CS760 Machine Learning

Neural Networks IV

Kirthi Kandasamy

University of Wisconsin-Madison

March 6, 2023

Announcements

- **Midterm**
    - Alternative date on 3/21. You should have received an email from me.
    - Please do not share answers after you finish your midterm.

- **Midterm course evaluations**

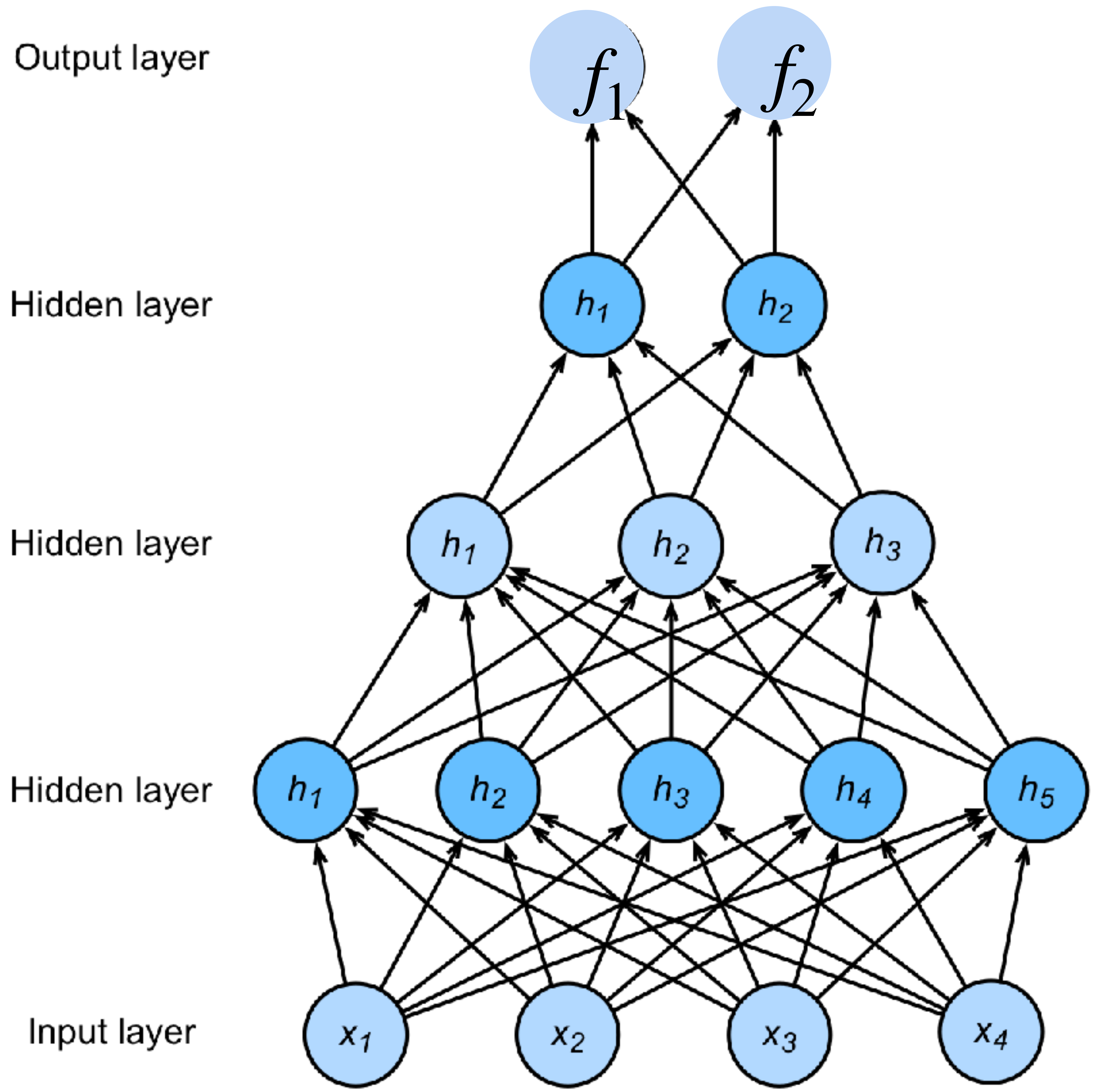
Outline

- **Convolutional operations**
    - 2D convolution
    - Padding, stride etc
    - Multiple input and output channels
    - Pooling
- **Convolutional Neural Networks & CNN Architectures**

Outline

- **Convolutional operations**
    - 2D convolution
    - Padding, stride etc
    - Multiple input and output channels
    - Pooling
- **Convolutional Neural Networks & CNN Architectures**

Multi-layer perceptrons



$$\mathbf{h}_1 = \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$$
$$\mathbf{h}_2 = \sigma(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2)$$
$$\mathbf{h}_3 = \sigma(\mathbf{W}_3\mathbf{h}_2 + \mathbf{b}_3)$$
$$\mathbf{f} = \mathbf{W}_4\mathbf{h}_3 + \mathbf{b}_4$$
$$\mathbf{y} = \text{softmax}(\mathbf{f})$$

NNs are composition of nonlinear functions

Classifying Images

How to classify

Cats vs. dogs?

Dual

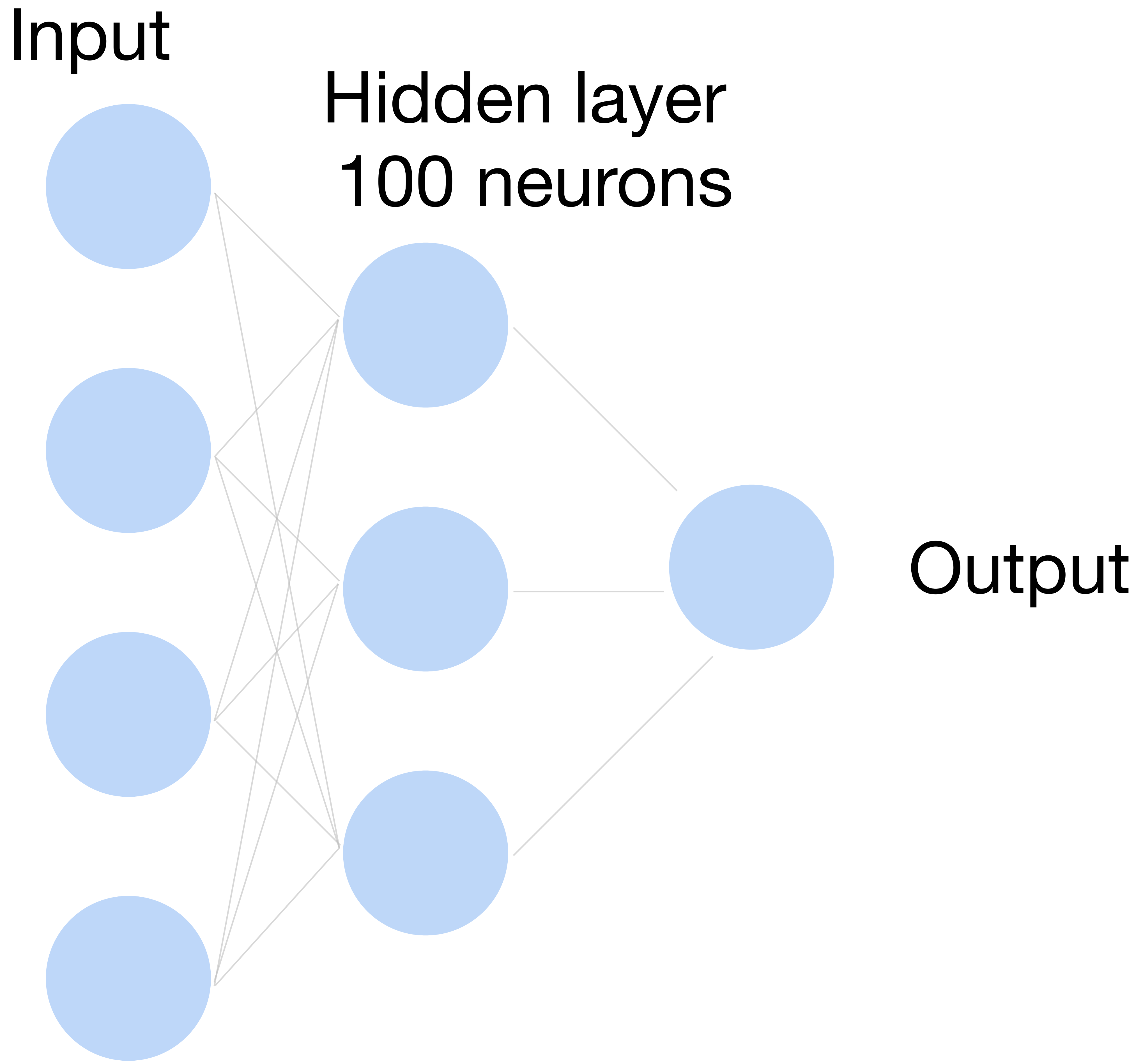
12MP

wide-angle and
telephoto cameras

36M floats in a RGB image!

Classifying Images with fully connected NNs

Cats vs. dogs?



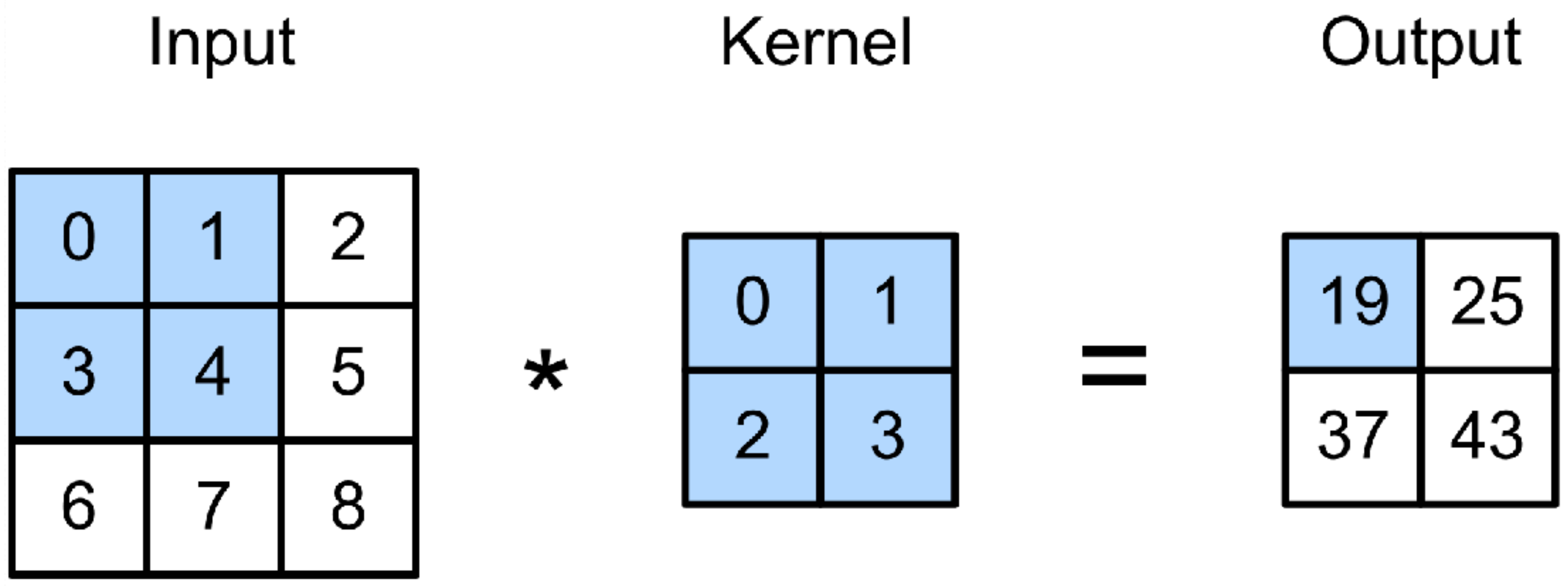
~ 36M elements x 100 = ~**3.6B** parameters!

Convolutions come to rescue!

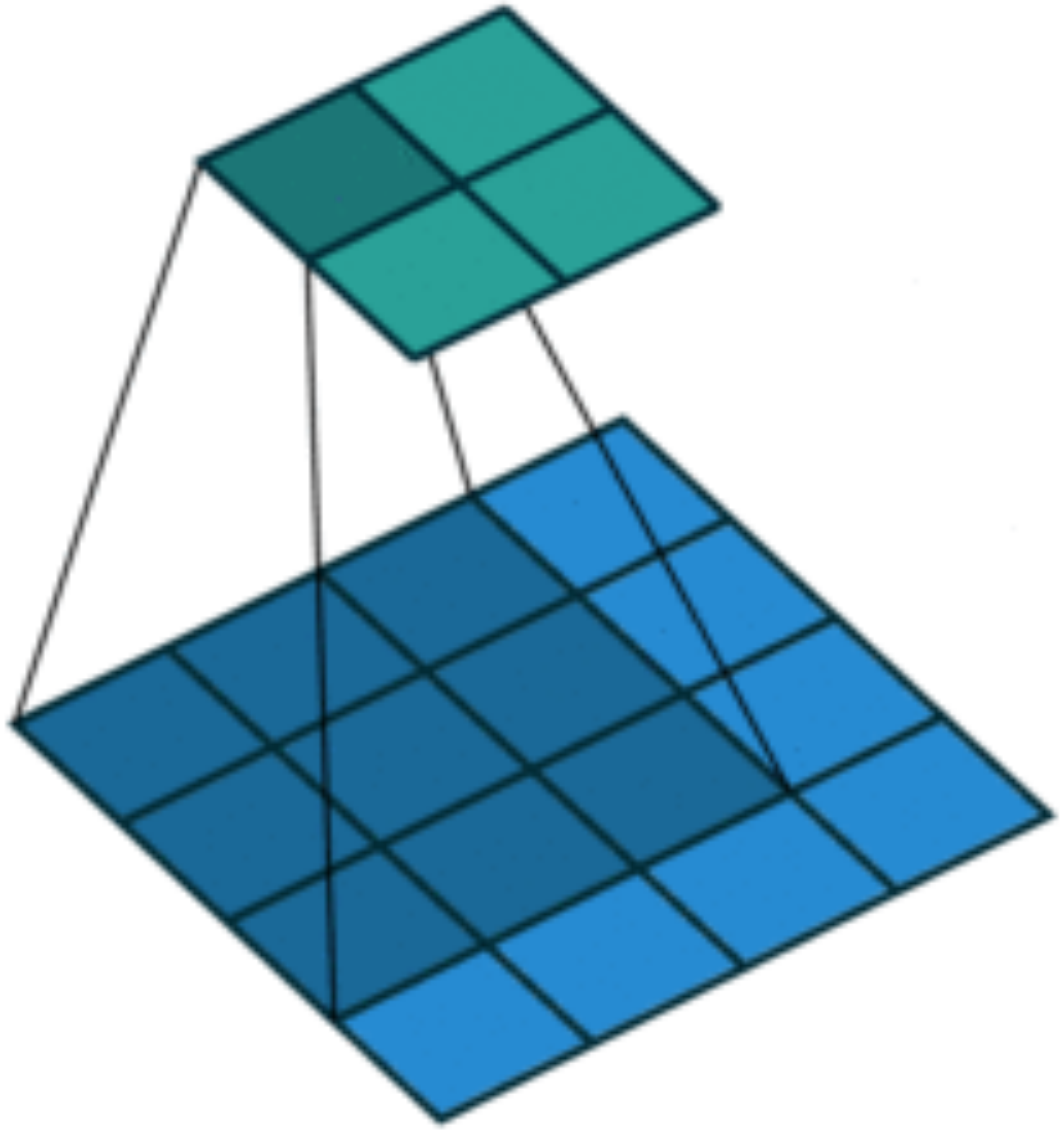
Why Convolution?

- Reduces number of parameters
- Translation Invariance
- Locality

2-D Convolution



$$
\begin{aligned}
0\times 0+1\times 1+3\times 2+4\times 3 &= 19,\\
1\times 0+2\times 1+4\times 2+5\times 3 &= 25,\\
3\times 0+4\times 1+6\times 2+7\times 3 &= 37,\\
4\times 0+5\times 1+7\times 2+8\times 3 &= 43.
\end{aligned}
$$



(vdumoulin@ Github)

2-D Convolution Layer

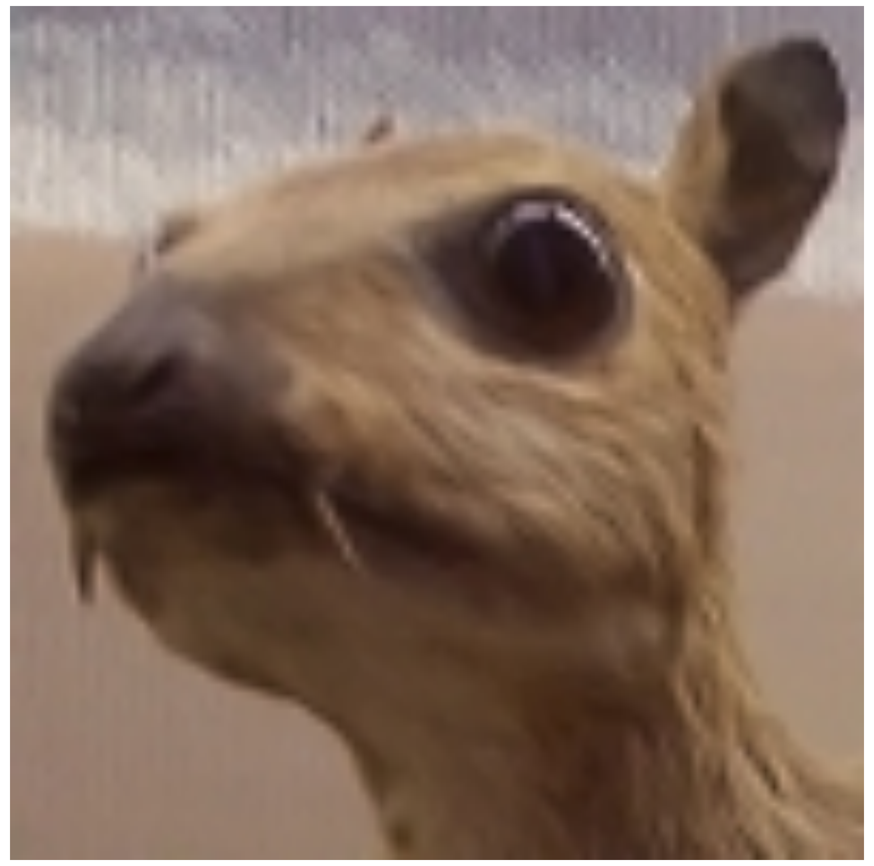
$$\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} = \begin{bmatrix} 19 & 25 \\ 37 & 43 \end{bmatrix}$$

- $\mathbf{X} : n_h \times n_w$ input matrix
- $\mathbf{W} : k_h \times k_w$ kernel matrix
- b: scalar bias
- $\mathbf{Y} : (n_h - k_h + 1) \times (n_w - k_w + 1)$ output matrix

$$\mathbf{Y} = \mathbf{X} \star \mathbf{W} + b$$

- **W** and *b* are learnable parameters

Examples

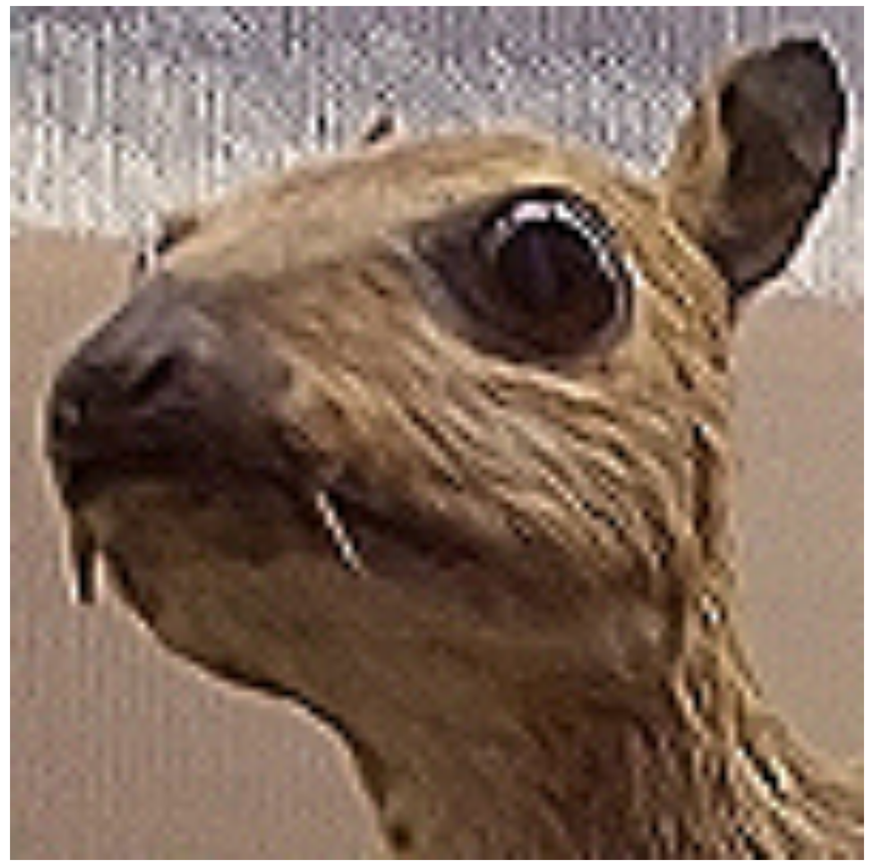


(wikipedia)

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Edge Detection

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$



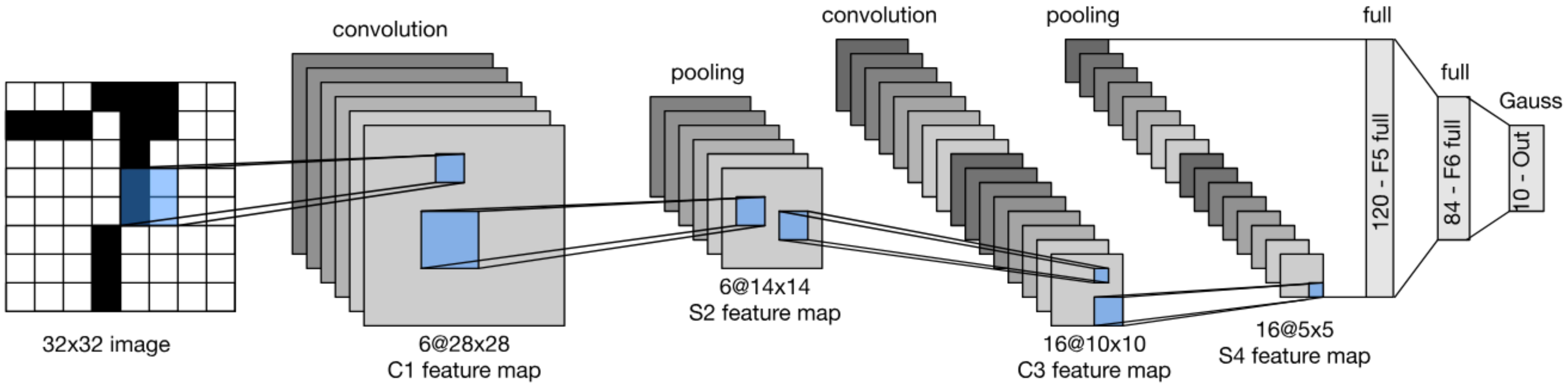
Sharpen

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Gaussian Blur

Convolutional Neural Networks

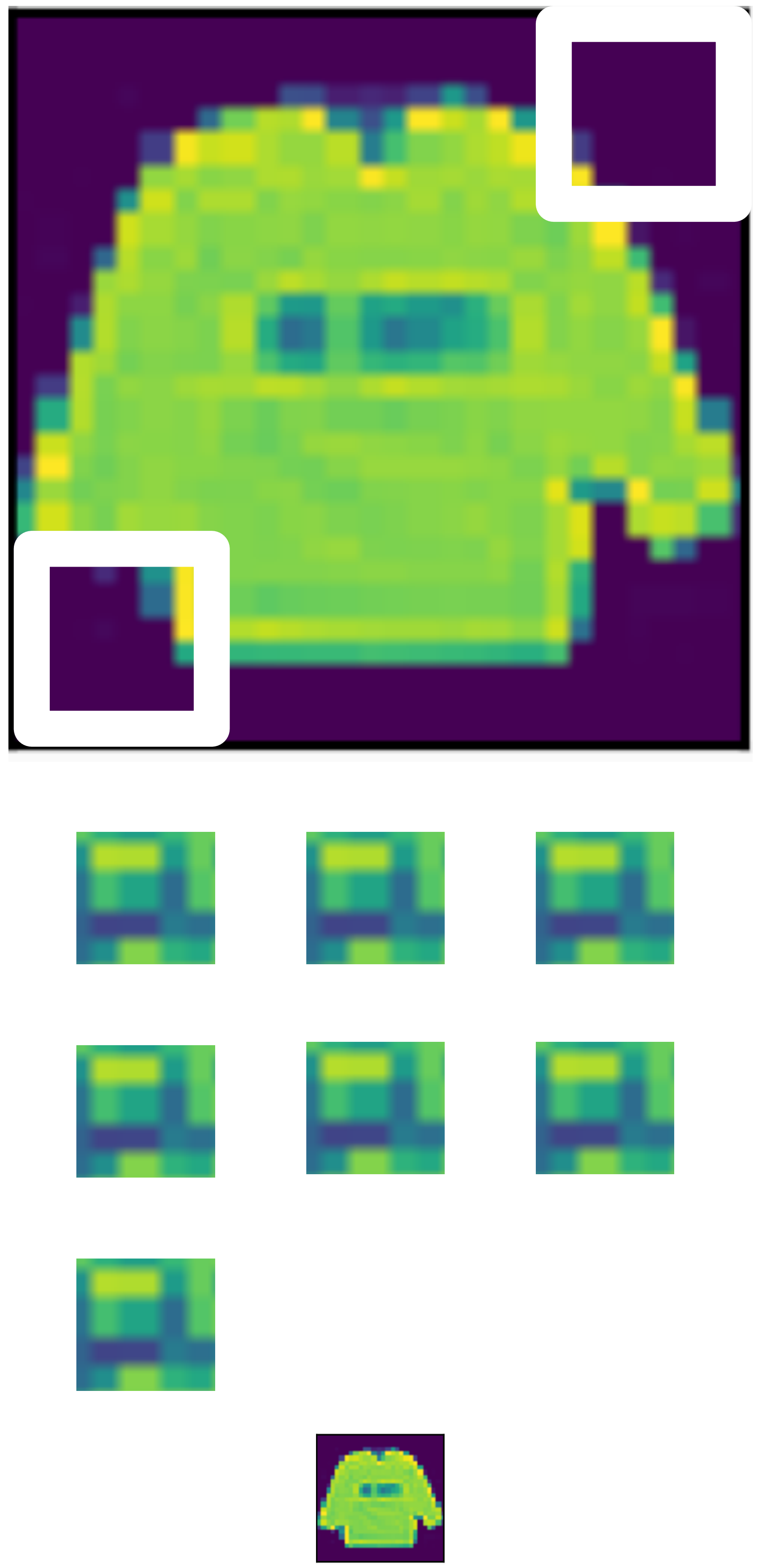
Convolutional networks: neural networks that use convolution in place of general matrix multiplication in at least one of their layers



Padding

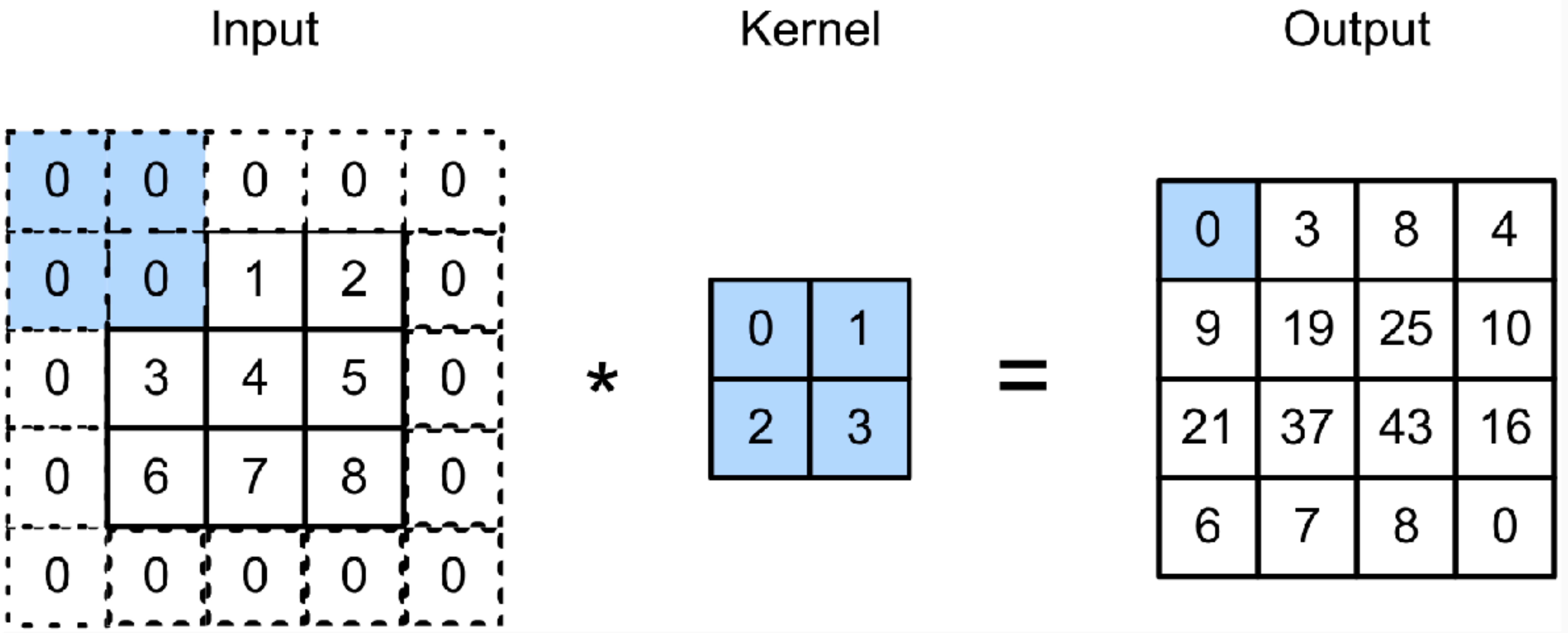
- Given a 32 x 32 input image
- Apply convolution with 5 x 5 kernel
  - 28 x 28 output with 1 layer
  - 4 x 4 output with 7 layers
- Shape decreases faster with larger kernels

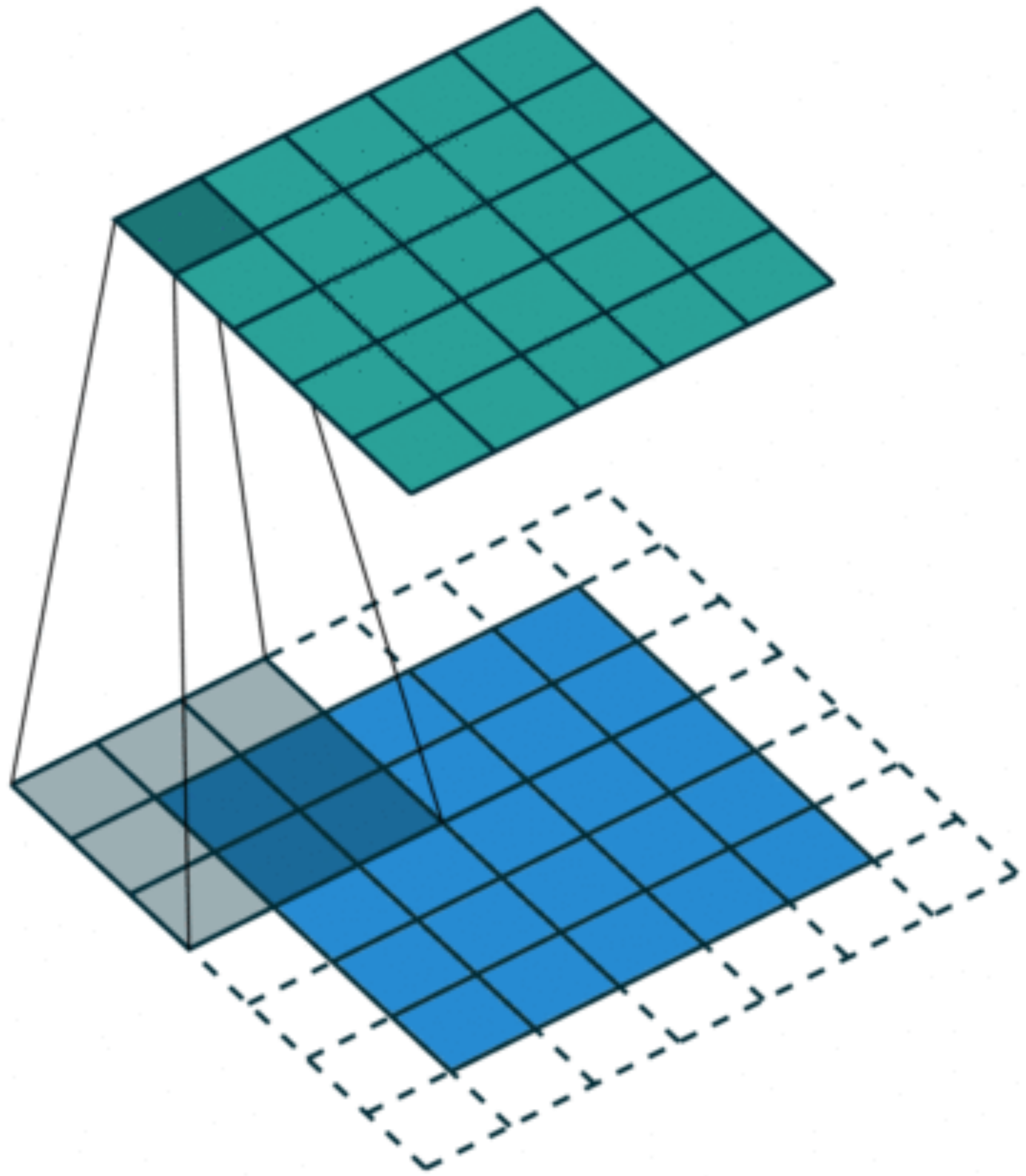
- Padding preserves **edge information!**



Padding

Padding adds rows/columns around input





$$0 \times 0 + 0 \times 1 + 0 \times 2 + 0 \times 3 = 0$$

Padding

- Padding p_h rows and p_w columns, output shape will be

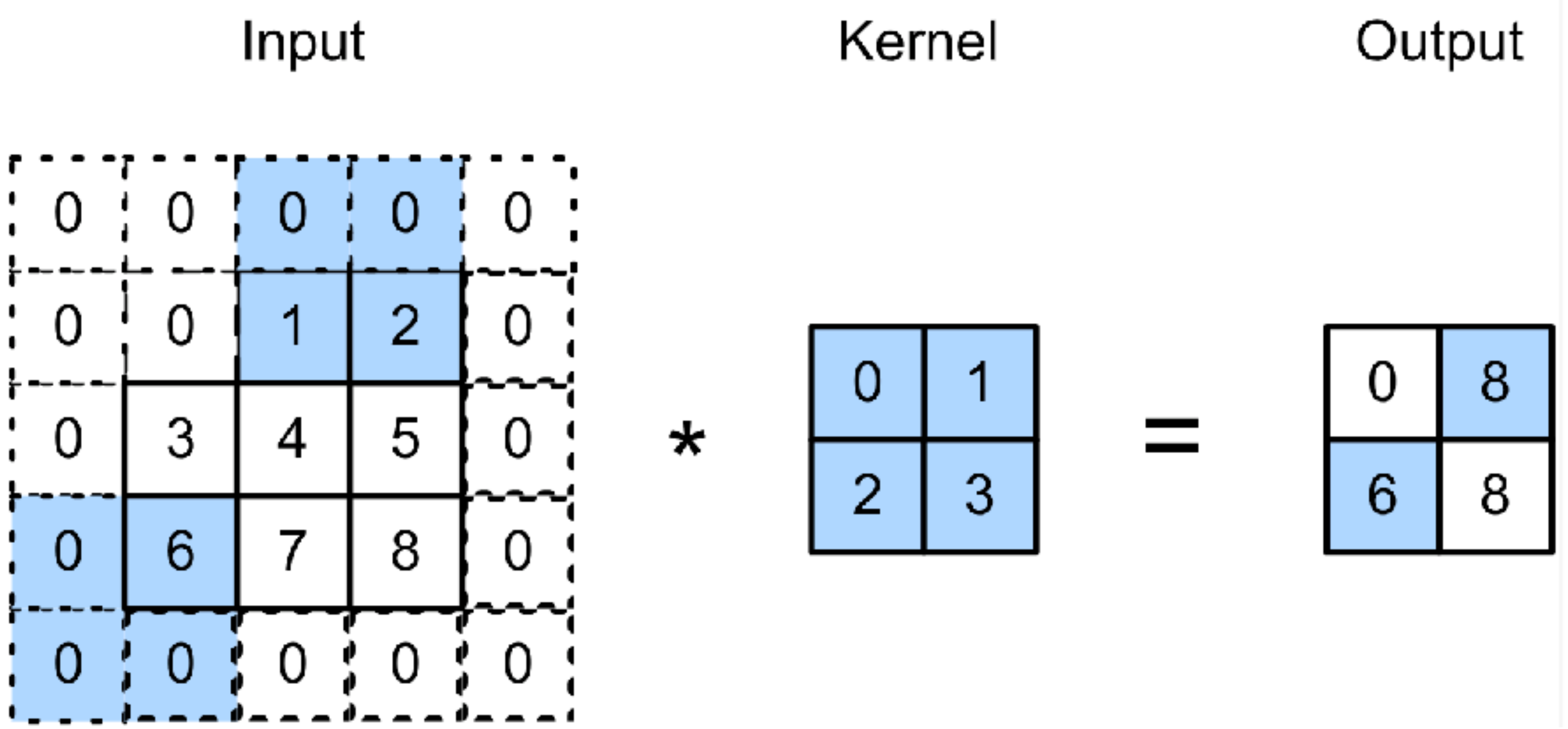
$$(n_h - k_h + p_h + 1) \times (n_w - k_w + p_w + 1)$$

- A common choice is $p_h = k_h - 1$ and $p_w = k_w - 1$
  - Odd k_h : pad $p_h/2$ on both sides
  - Even k_h : pad $\lceil p_h/2 \rceil$ on top, $\lfloor p_h/2 \rfloor$ on bottom

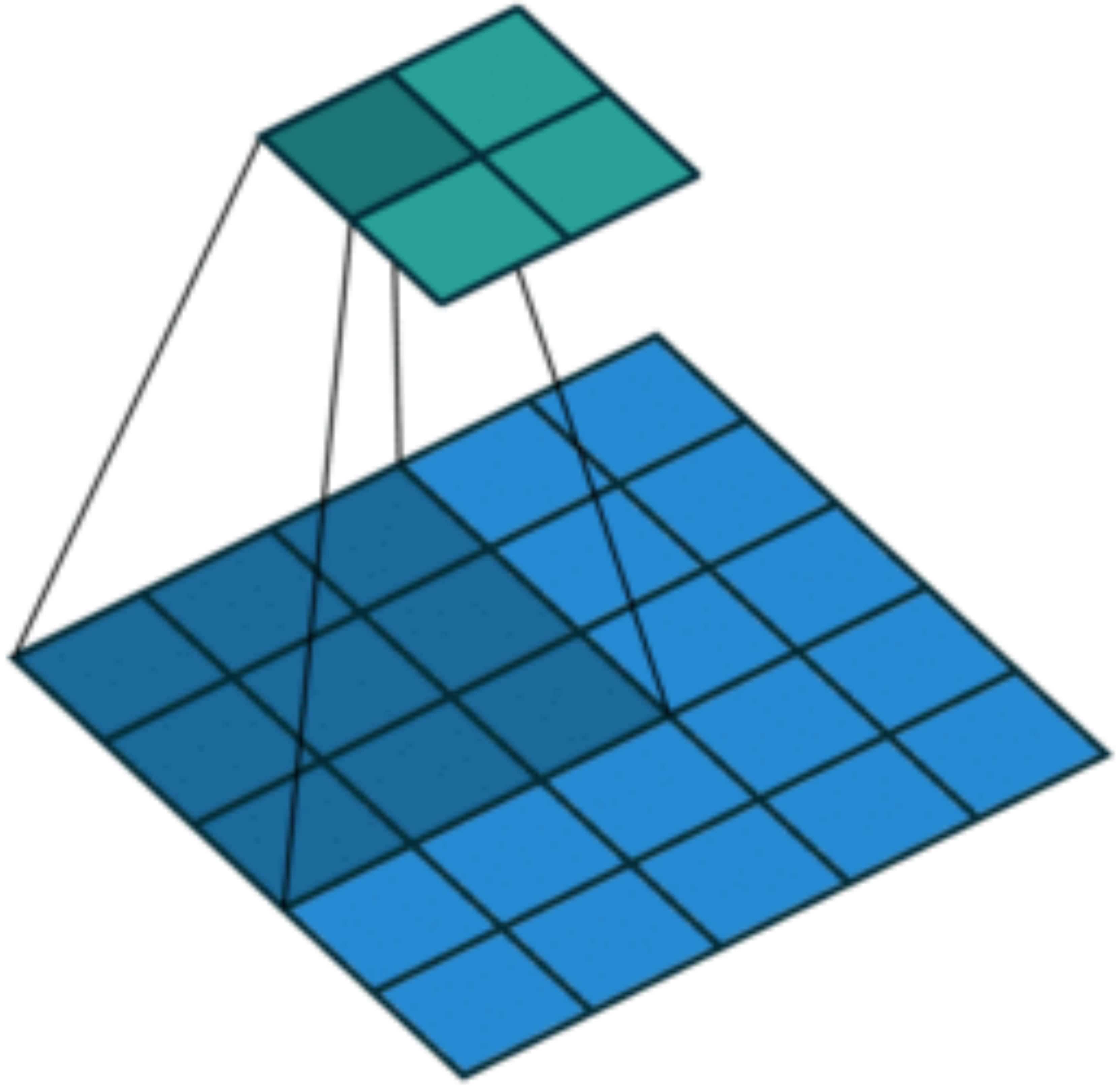
Stride

- Stride is the #rows/#columns per slide

Strides of 3 and 2 for height and width



$$0 \times 0 + 0 \times 1 + 1 \times 2 + 2 \times 3 = 8$$
$$0 \times 0 + 6 \times 1 + 0 \times 2 + 0 \times 3 = 6$$



Stride

- Given stride s_h for the height and stride s_w for the width, the output shape is

$$\lfloor(n_h - k_h + p_h + s_h)/s_h\rfloor \times \lfloor(n_w - k_w + p_w + s_w)/s_w\rfloor$$

- With $p_h = k_h - 1$ and $p_w = k_w - 1$

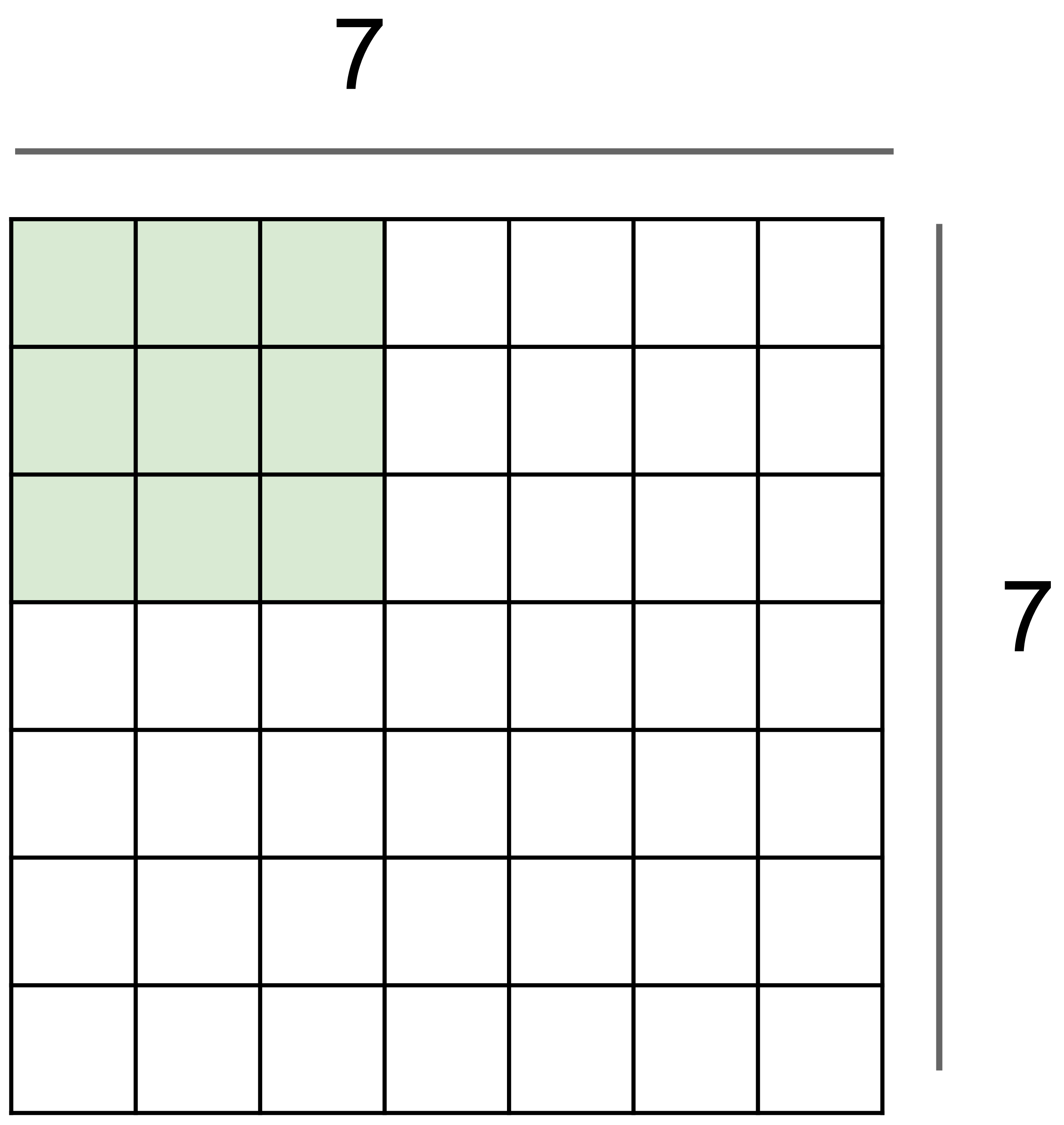
$$\lfloor(n_h + s_h - 1)/s_h\rfloor \times \lfloor(n_w + s_w - 1)/s_w\rfloor$$

- If input height/width are divisible by strides

$$(n_h/s_h) \times (n_w/s_w)$$

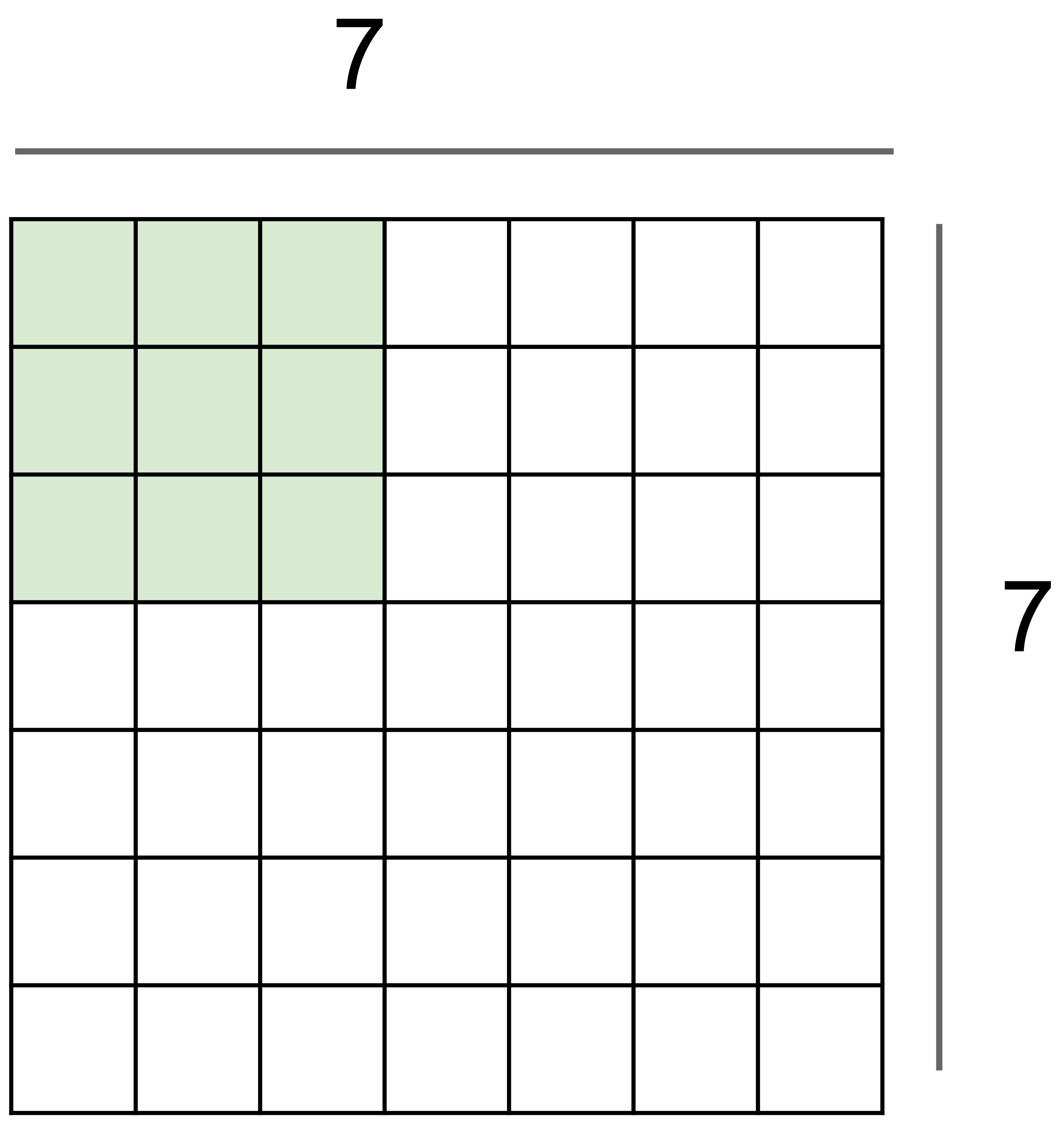
Q1. Suppose we want to perform convolution on a single channel image of size 7x7 (no padding) with a kernel of size 3x3, and stride = 2. What is the dimension of the output?

A. 3x3
B. 7x7
C. 5x5
D. 2x2



Q1. Suppose we want to perform convolution on a single channel image of size 7x7 (no padding) with a kernel of size 3x3, and stride = 2. What is the dimension of the output?

A. 3x3
B. 7x7
C. 5x5
D. 2x2



$$\lfloor (n_h - k_h + p_h + s_h)/s_h \rfloor \times \lfloor (n_w - k_w + p_w + s_w)/s_w \rfloor$$

Multiple Input and Output Channels

Multiple Input Channels

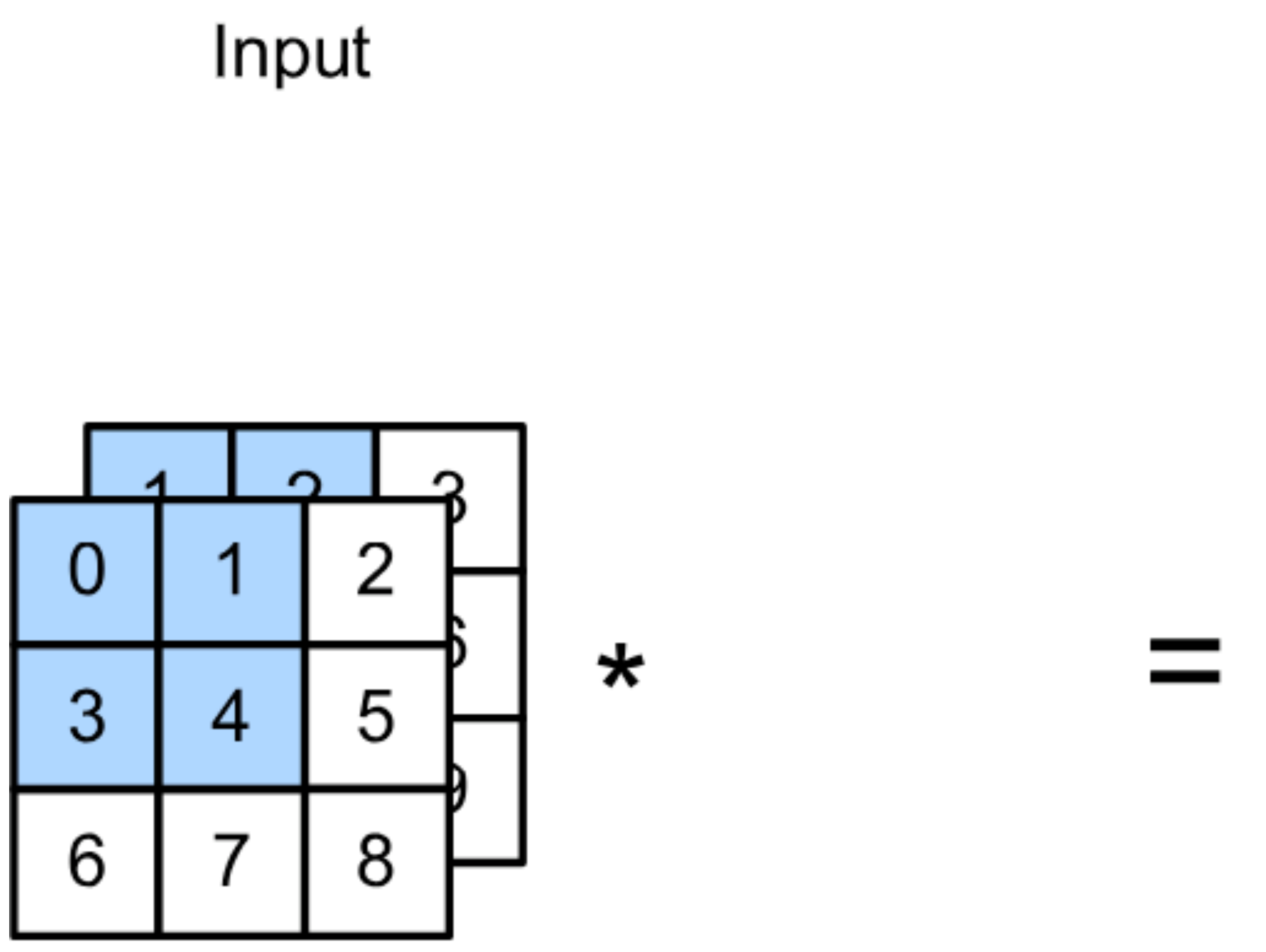
- Color image may have three RGB channels

Multiple Input Channels

- Color image may have three RGB channels

Multiple Input Channels

- Have a kernel for each channel, and then sum results over channels



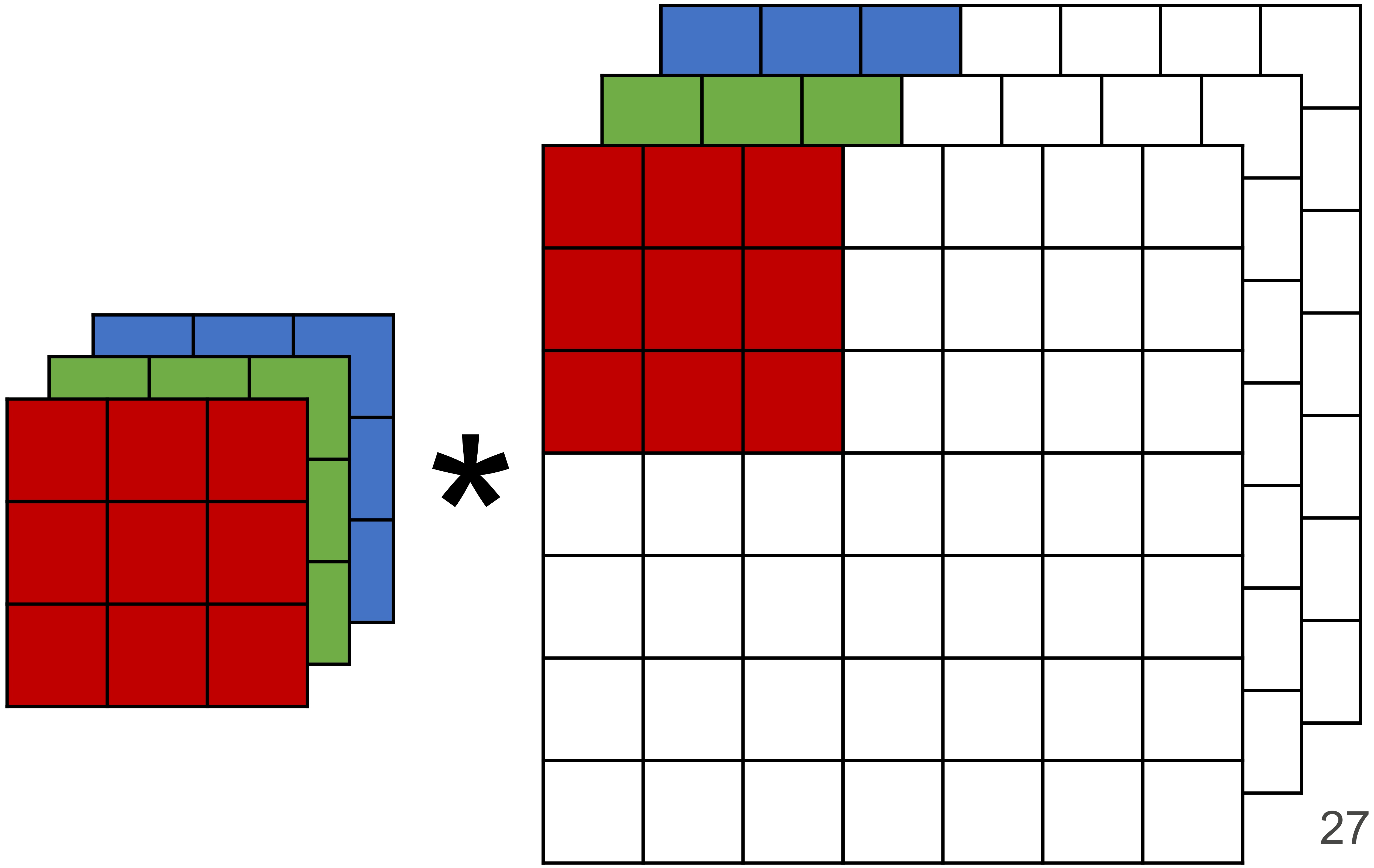
Multiple Input Channels

- $\mathbf{X} : c_i \times n_h \times n_w$ input
- $\mathbf{W} : c_i \times k_h \times k_w$ kernel
- $\mathbf{Y} : m_h \times m_w$ output

$$\mathbf{Y} = \mathbf{X} \star \mathbf{W} = \sum_{i=0}^{c_i} \mathbf{X}_{i,:,:} \star \mathbf{W}_{i,:,:} + b$$

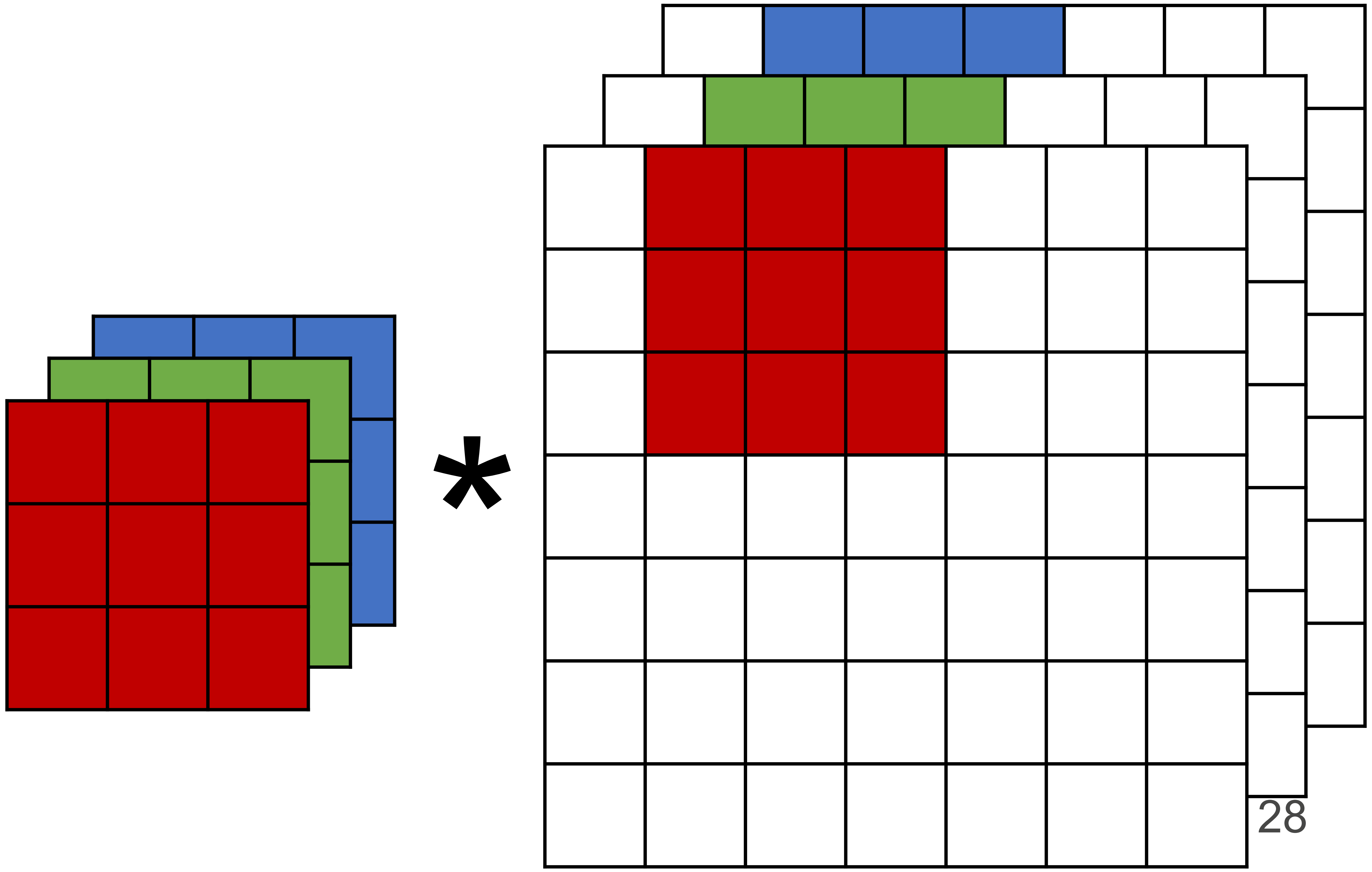
Multiple Input Channels

- RGB images have 3 channels



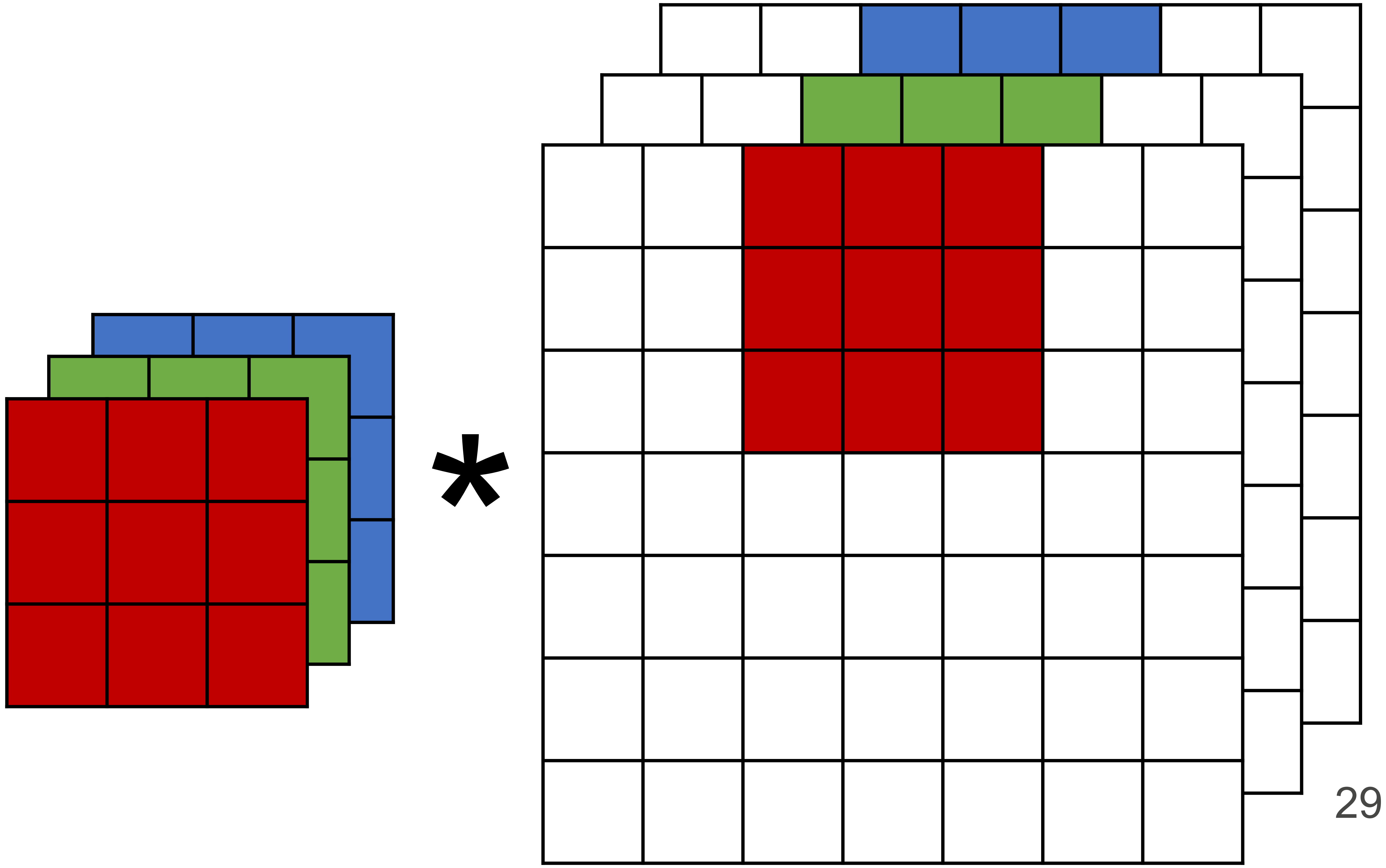
Multiple Input Channels

- RGB images have 3 channels



Multiple Input Channels

- RGB images have 3 channels



Multiple Output Channels

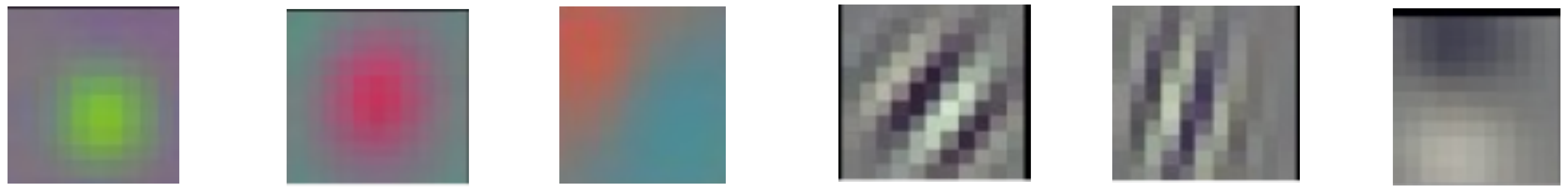
- We can have **multiple 3-D kernels**, each one generates an output channel
- Input $\mathbf{X} : c_i \times n_h \times n_w$
- Kernel $\mathbf{W} : c_o \times c_i \times k_h \times k_w$
- Output $\mathbf{Y} : c_o \times m_h \times m_w$

$$\mathbf{Y}_{i,:,:} = \mathbf{X} \star \mathbf{W}_{i,:,:,:} + b$$

$$\text{for } i = 1, \ldots, c_o$$

Multiple Input/Output Channels

- Each 3-D kernel may recognize a particular pattern



(Gabor filters)

Q. Suppose we want to perform convolution on a RGB image of size 224x224 (no padding) with 64 kernels of size 3x3. Stride = 1. What is a reasonable estimate of the total number of scalar multiplications involved in this operation (without considering any optimization in matrix multiplication)?

A. 64x3x3x222x222

B. 64x3x3x222

C. 3x3x222x222

D. 64x3x3x3x222x222

Q3. Suppose we want to perform convolution on a RGB image of size 224x224 (no padding) with 64 kernels of size 3x3. Stride = 1. What is a reasonable estimate of the total number of scalar multiplications involved in this operation (without considering any optimization in matrix multiplication)?

A. 64x3x3x222x222

B. 64x3x3x222

C. 3x3x222x222

D. 64x3x3x3x222x222

Q. Suppose we want to perform convolution on a RGB image of size 224x224 (no padding) with 64 kernels of size 3x3. Stride = 1. Which is a reasonable estimate of the total number of learnable parameters?

A. 64x222x222

B. 64x3x3x222

C. 3x3x3x64

D. (3x3x3+1)x64

Q. Suppose we want to perform convolution on a RGB image of size 224x224 (no padding) with 64 kernels of size 3x3. Stride = 1. Which is a reasonable estimate of the total number of learnable parameters?

A. 64x222x222

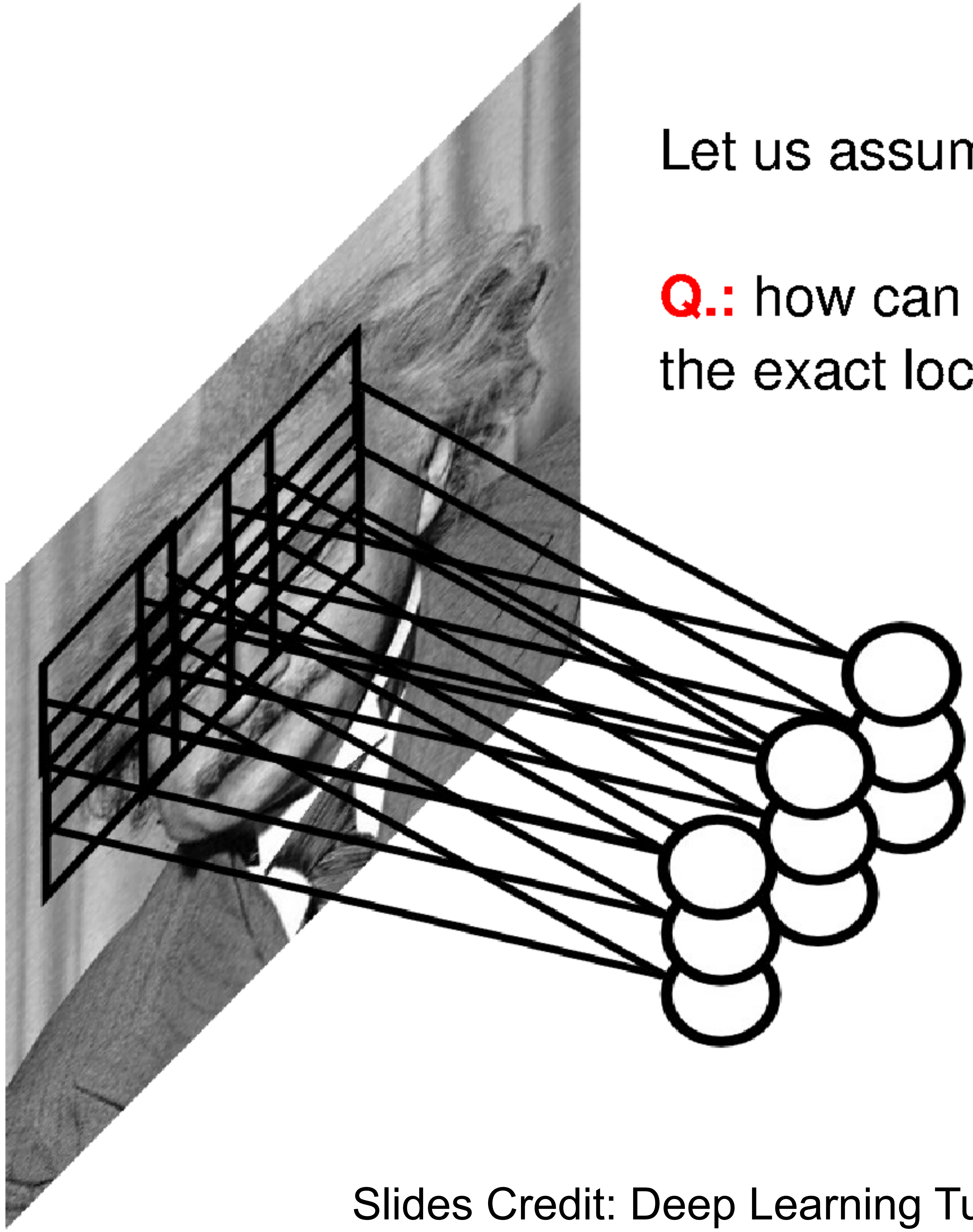
B. 64x3x3x222

C. 3x3x3x64

D. (3x3x3+1)x64

Pooling Layer

Pooling

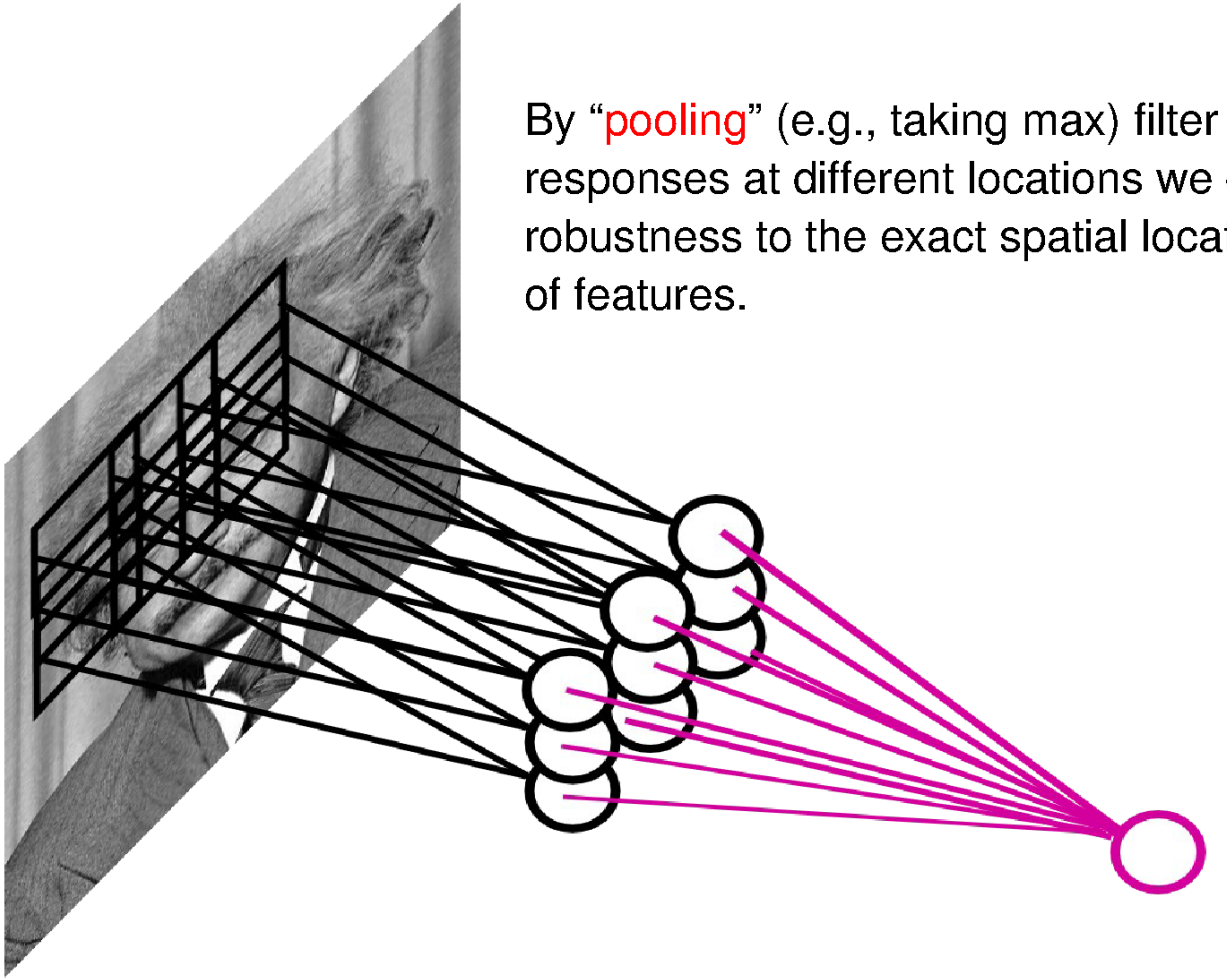


Let us assume filter is an “eye” detector.

Q.: how can we make the detection robust to the exact location of the eye?

Slides Credit: Deep Learning Tutorial by Marc’Aurelio Ranzato

Pooling

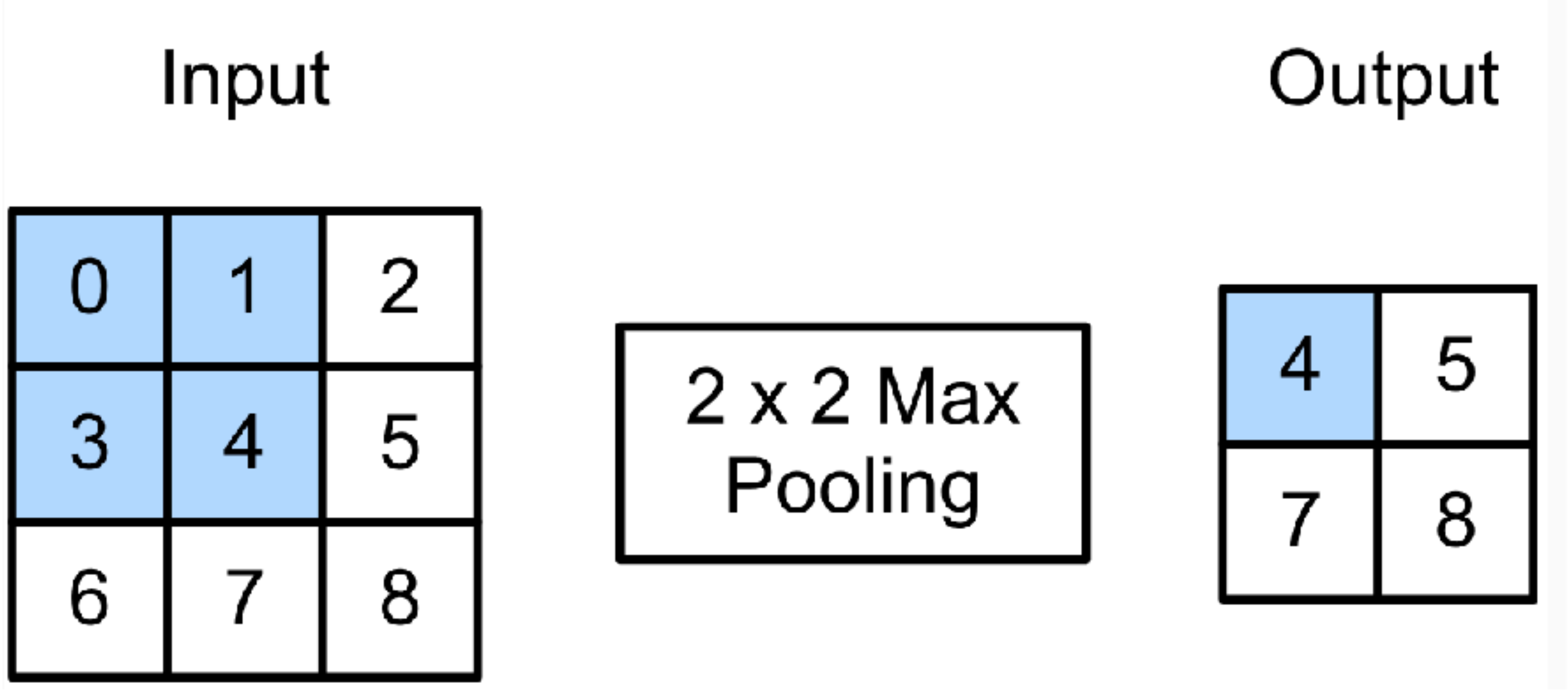


By "pooling" (e.g., taking max) filter responses at different locations we gain robustness to the exact spatial location of features.

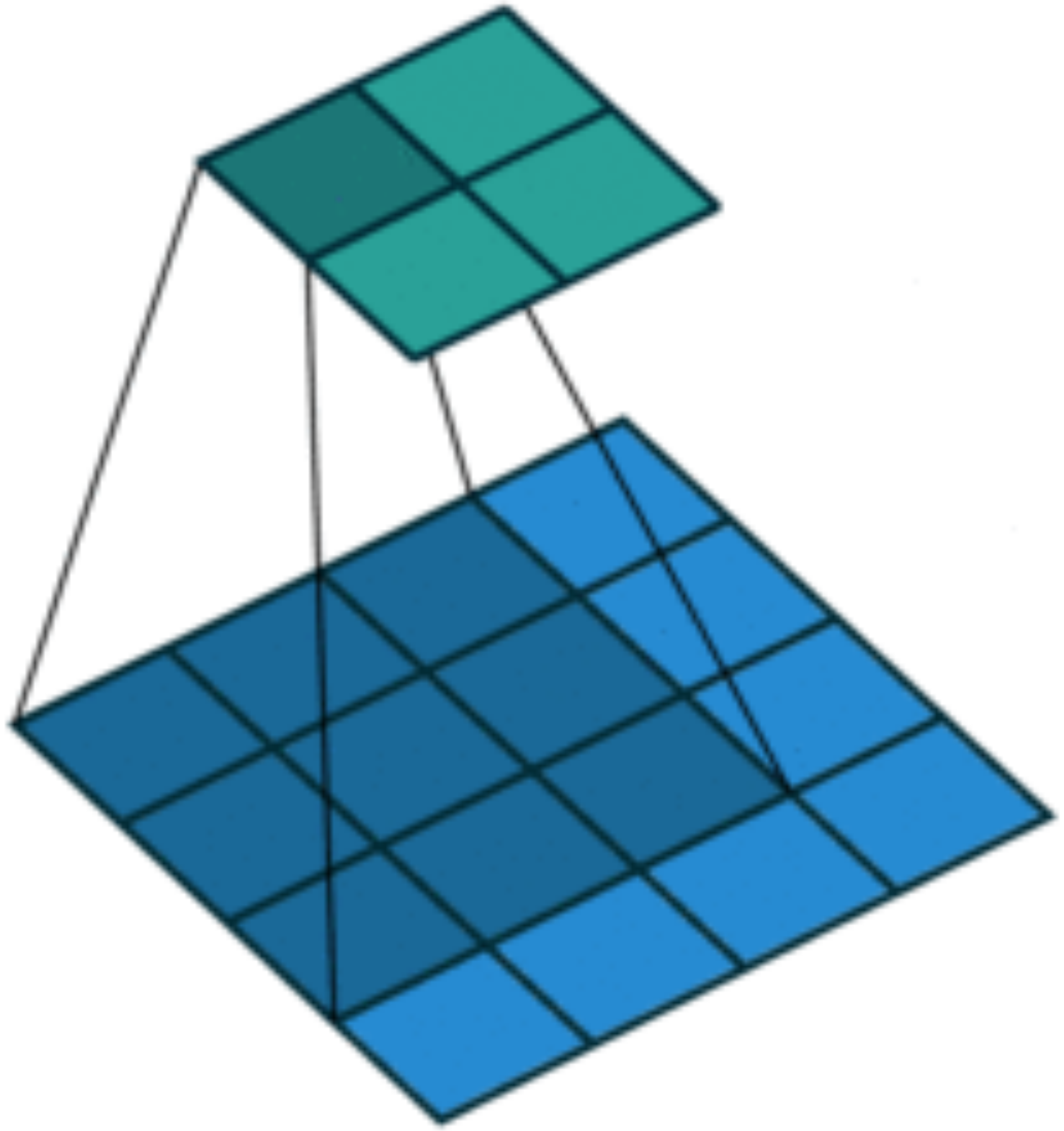
Slides Credit: Deep Learning Tutorial by Marc'Aurelio Ranzato

2-D Max Pooling

- Returns the maximal value in the sliding window



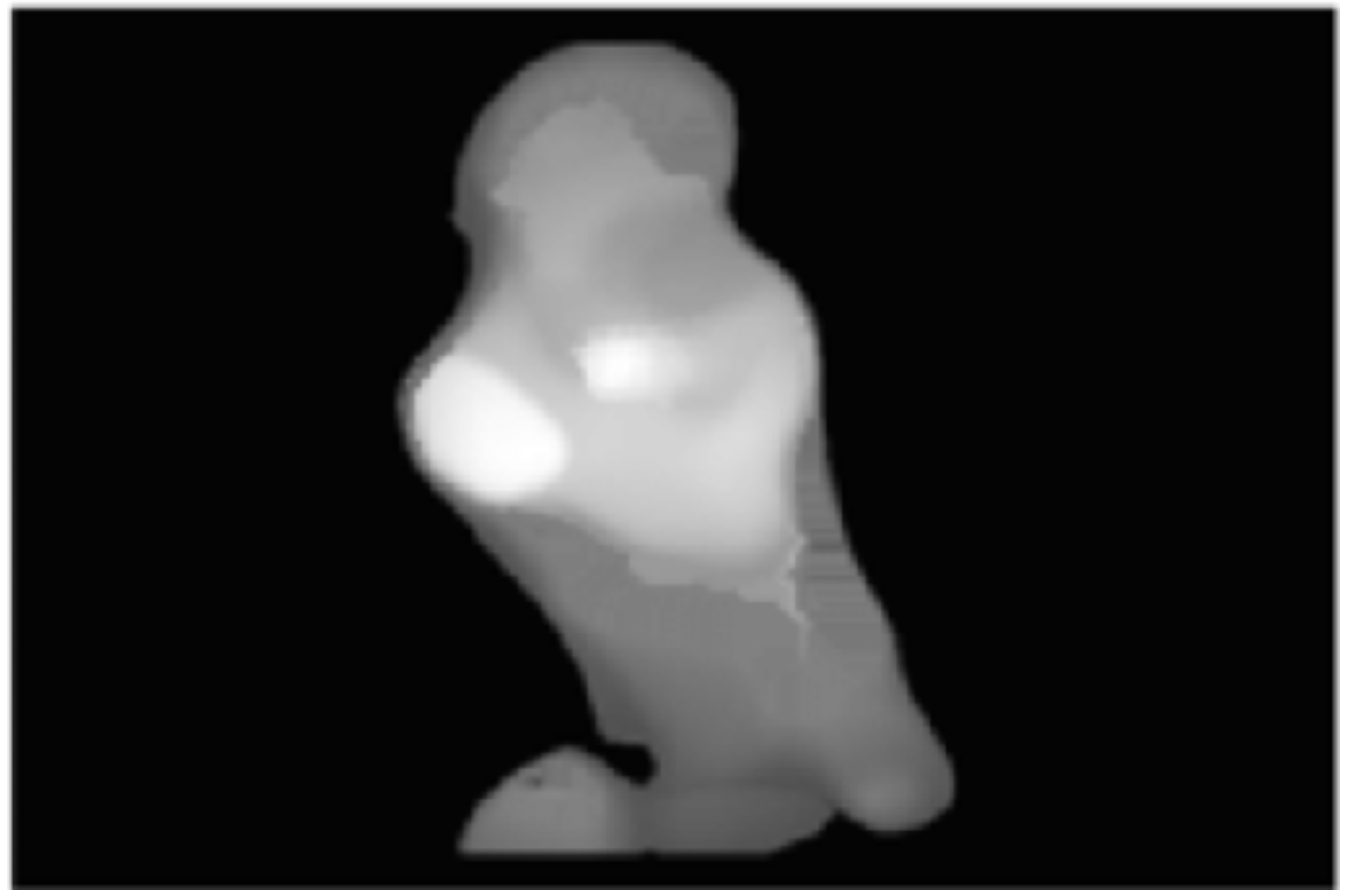
$$\max(0,1,3,4) = 4$$



Average Pooling

- Max pooling: the strongest pattern signal in a window
- Average pooling: replace max with mean in max pooling
  - The average signal strength in a window

Max pooling

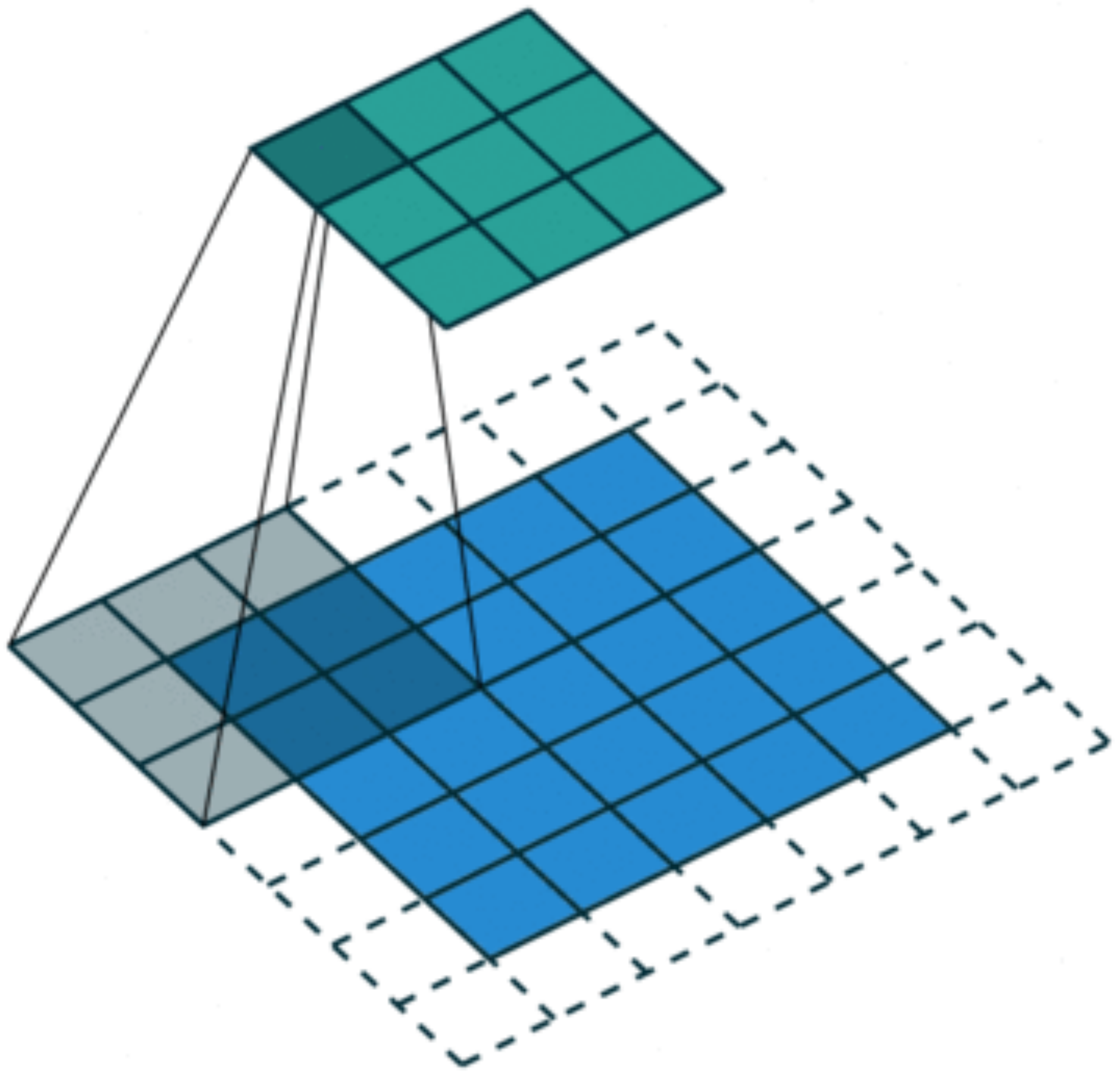


Average pooling

Padding, Stride, and Multiple Channels

- Pooling layers have similar padding and stride as convolutional layers
- No learnable parameters
- Apply pooling for each input channel to obtain the corresponding output channel

  #output channels = #input channels



Q. Suppose we want to perform 2x2 average pooling on the following single channel feature map of size 4x4 (no padding), and stride = 2. What is the output?

12	20	30	0
20	12	2	0
0	70	5	2
8	2	90	3

A.

20	30
70	90

B.

16	8
20	25

C.

20	30
20	25

D.

12	2
70	5

Q. Suppose we want to perform 2x2 average pooling on the following single channel feature map of size 4x4 (no padding), and stride = 2. What is the output?

12	20	30	0
20	12	2	0
0	70	5	2
8	2	90	3

A.

20	30
70	90

B.

16	8
20	25

C.

20	30
20	25

D.

12	2
70	5

Q. What is the output if we replace average pooling with 2 x 2 max pooling (other settings are the same)?

12	20	30	0
20	12	2	0
0	70	5	2
8	2	90	3

A.

20	30
70	90

B.

16	8
20	25

C.

20	30
20	25

D.

12	2
70	5

Q. What is the output if we replace average pooling with 2 x 2 max pooling (other settings are the same)?

12	20	30	0
20	12	2	0
0	70	5	2
8	2	90	3

A.

20	30
70	90

B.

16	8
20	25

C.

20	30
20	25

D.

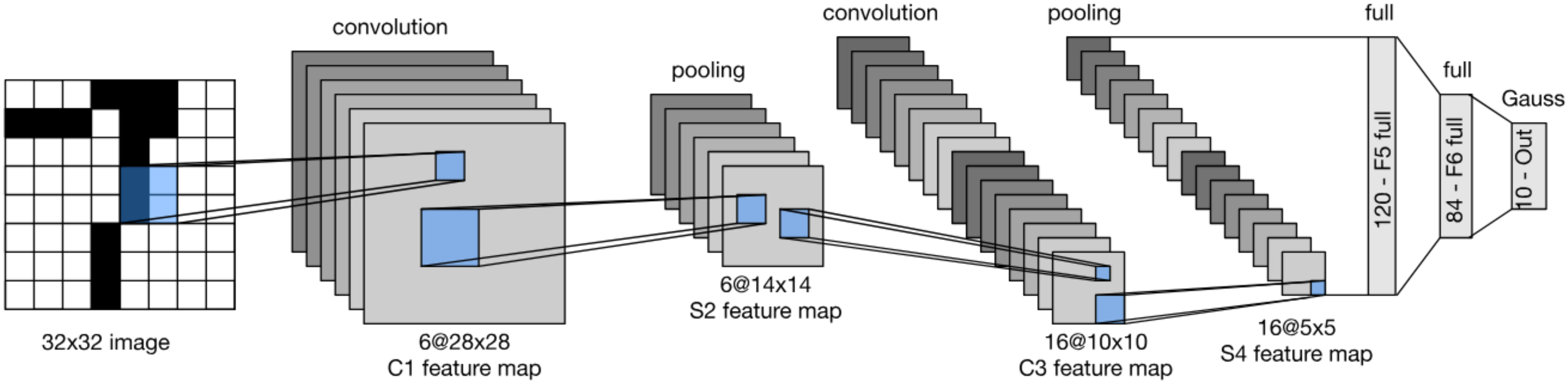
12	2
70	5

Outline

- **Convolutional operations**
    - 2D convolution
    - Padding, stride etc
    - Multiple input and output channels
    - Pooling
- **Convolutional Neural Networks & CNN Architectures**

Convolutional Neural Networks

Convolutional networks: neural networks that use convolution in place of general matrix multiplication in at least one of their layers



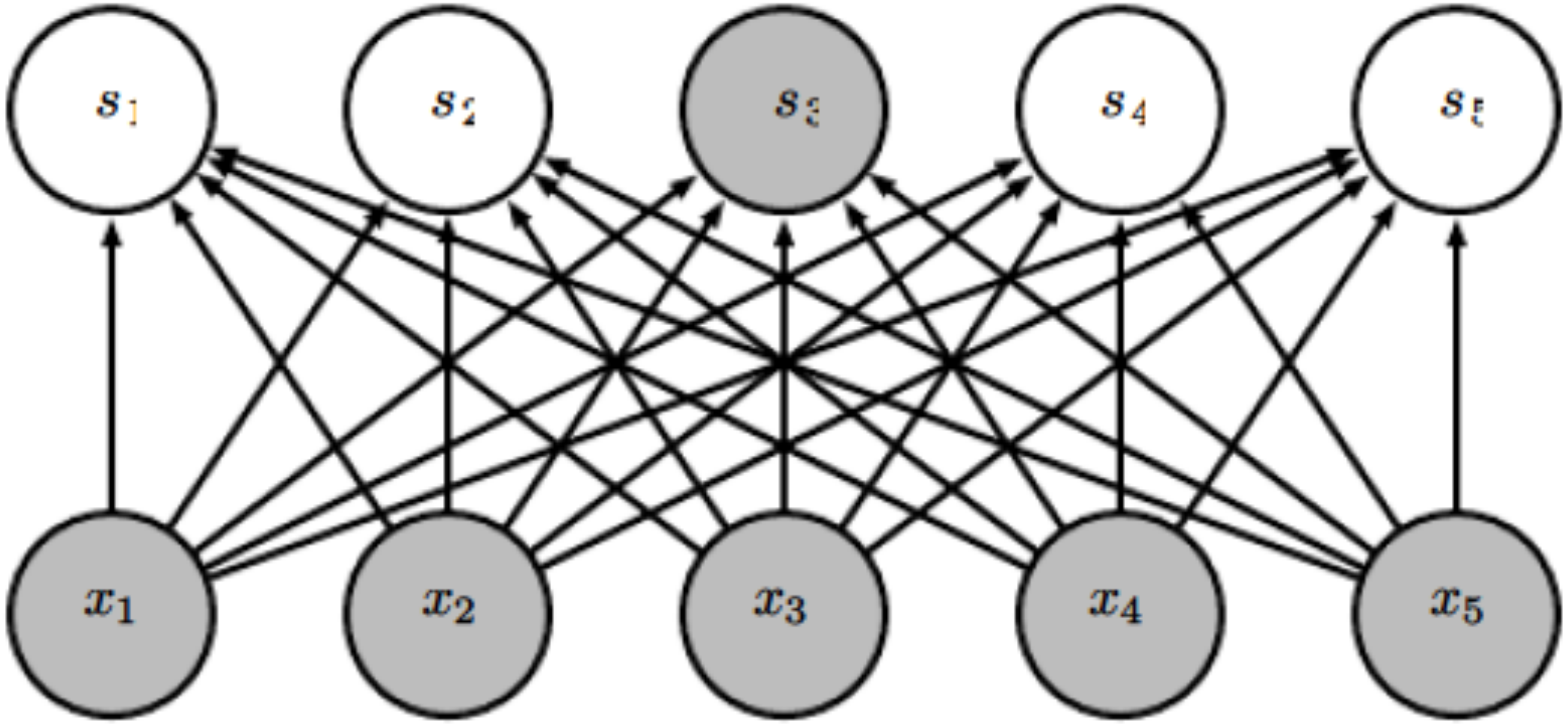
Why CNNs instead of MLPs?

- Translation Invariance
- Locality
- Reduces number of parameters

Why CNNs instead of MLPs?

Sparse interactions!

Fully connected layer, $m \times n$ edges



m output nodes

n input nodes

Figure from *Deep Learning*, by Goodfellow, Bengio, and Courville

Why CNNs instead of MLPs?

Sparse interactions!

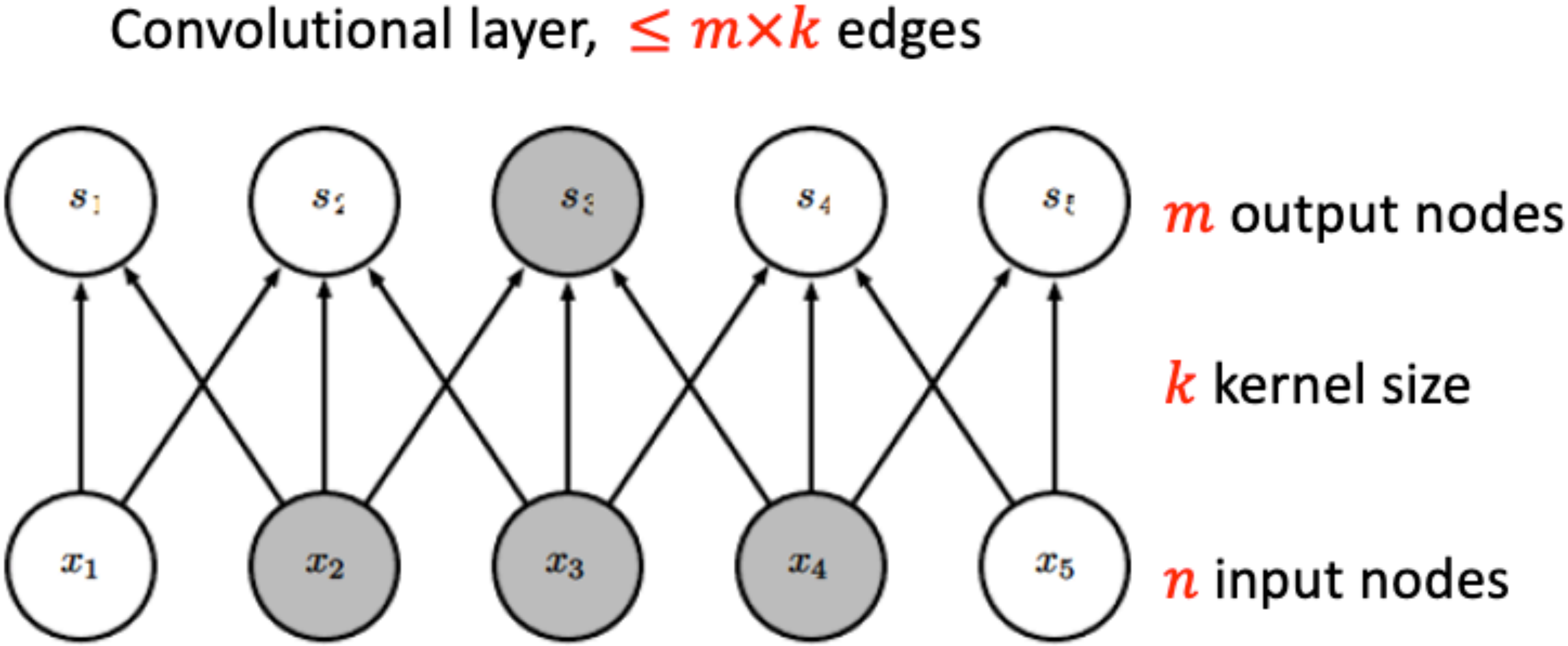
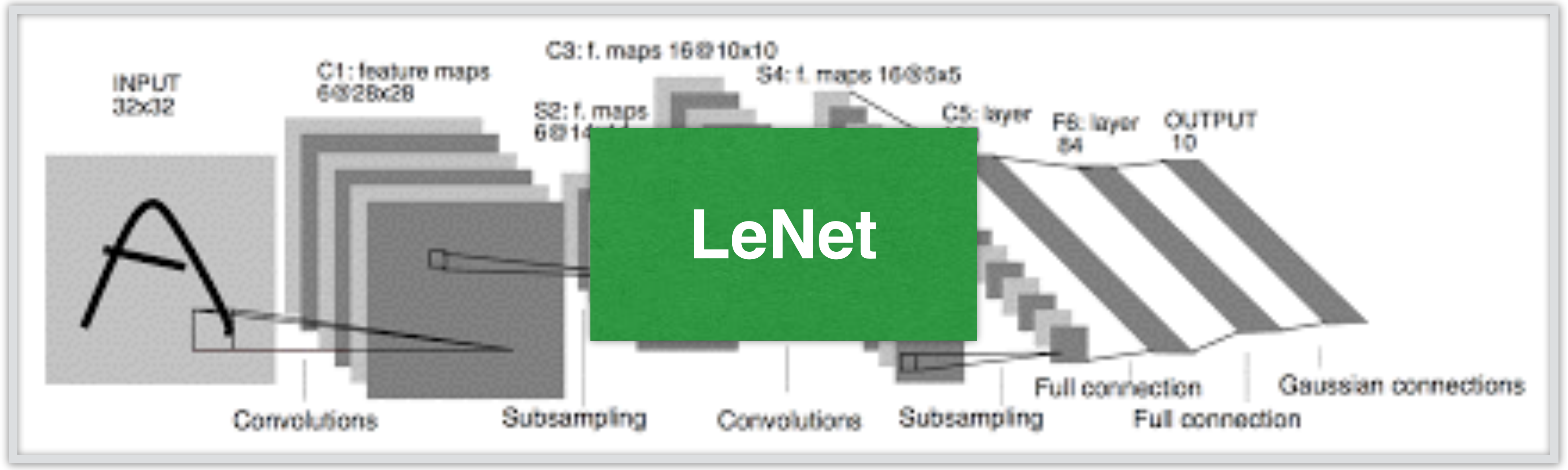
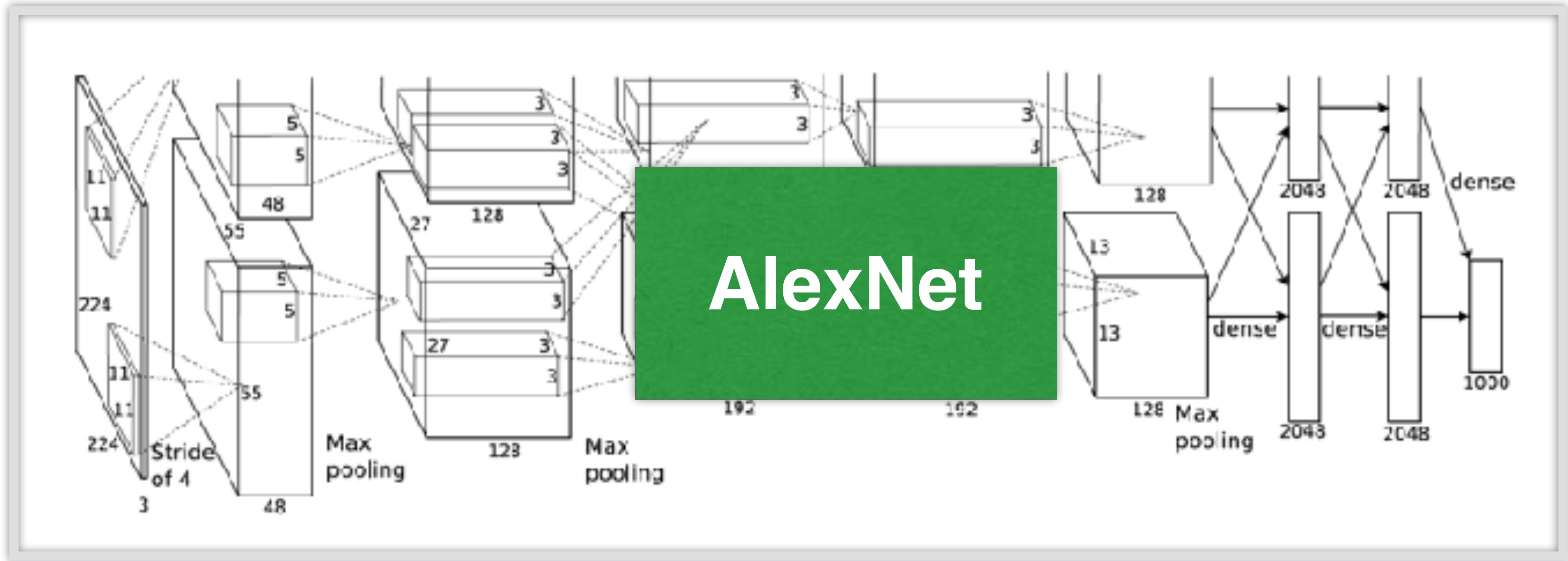
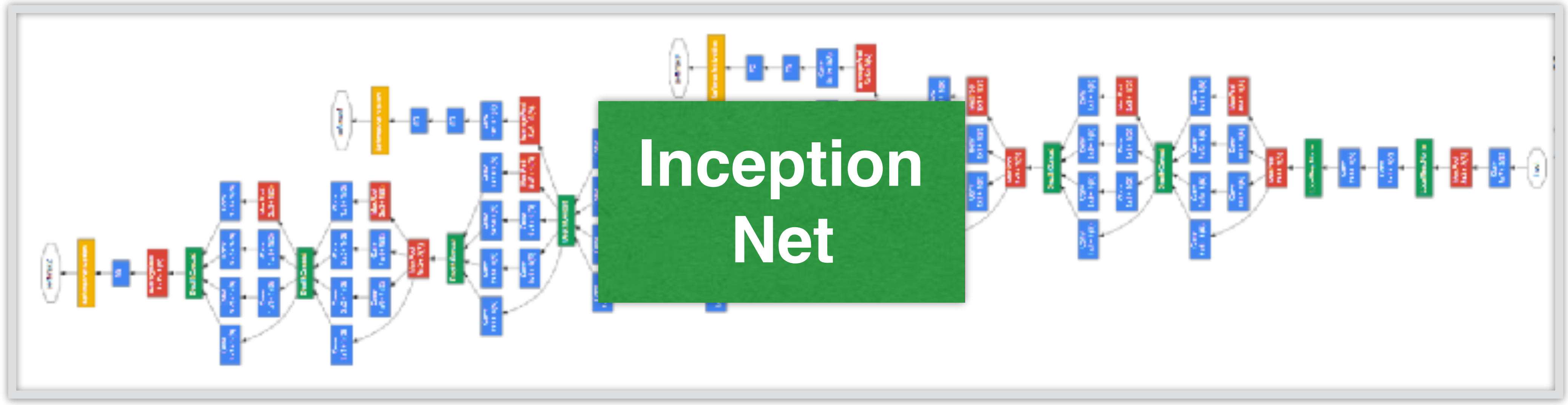


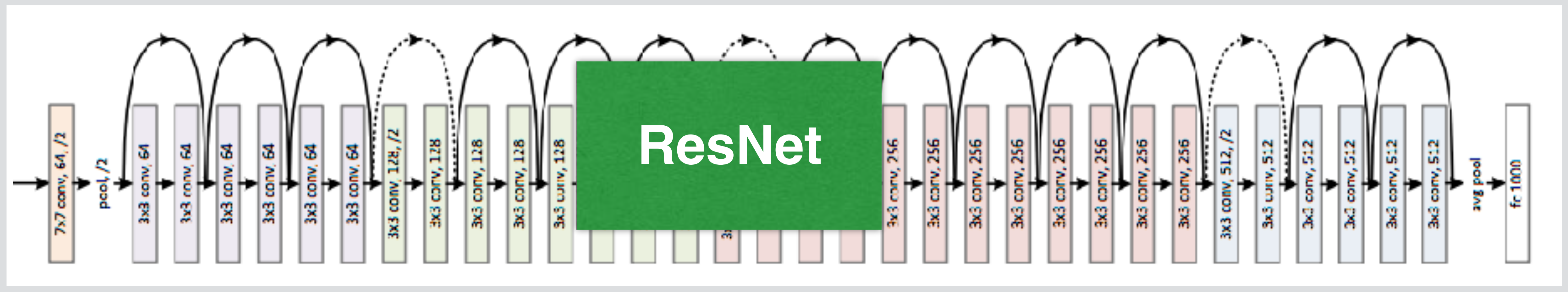
Figure from *Deep Learning*, by Goodfellow, Bengio, and Courville

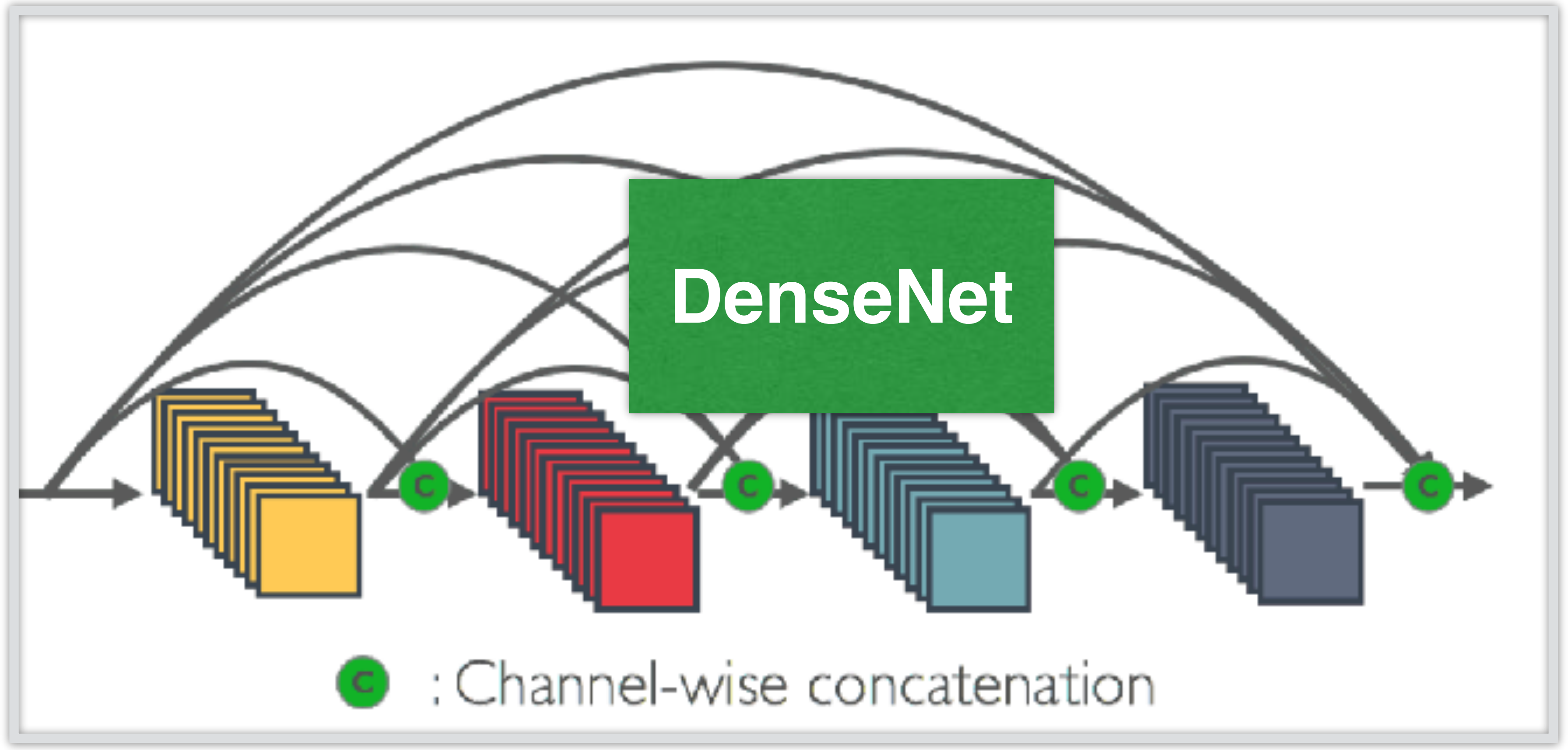
Evolution of neural net architectures







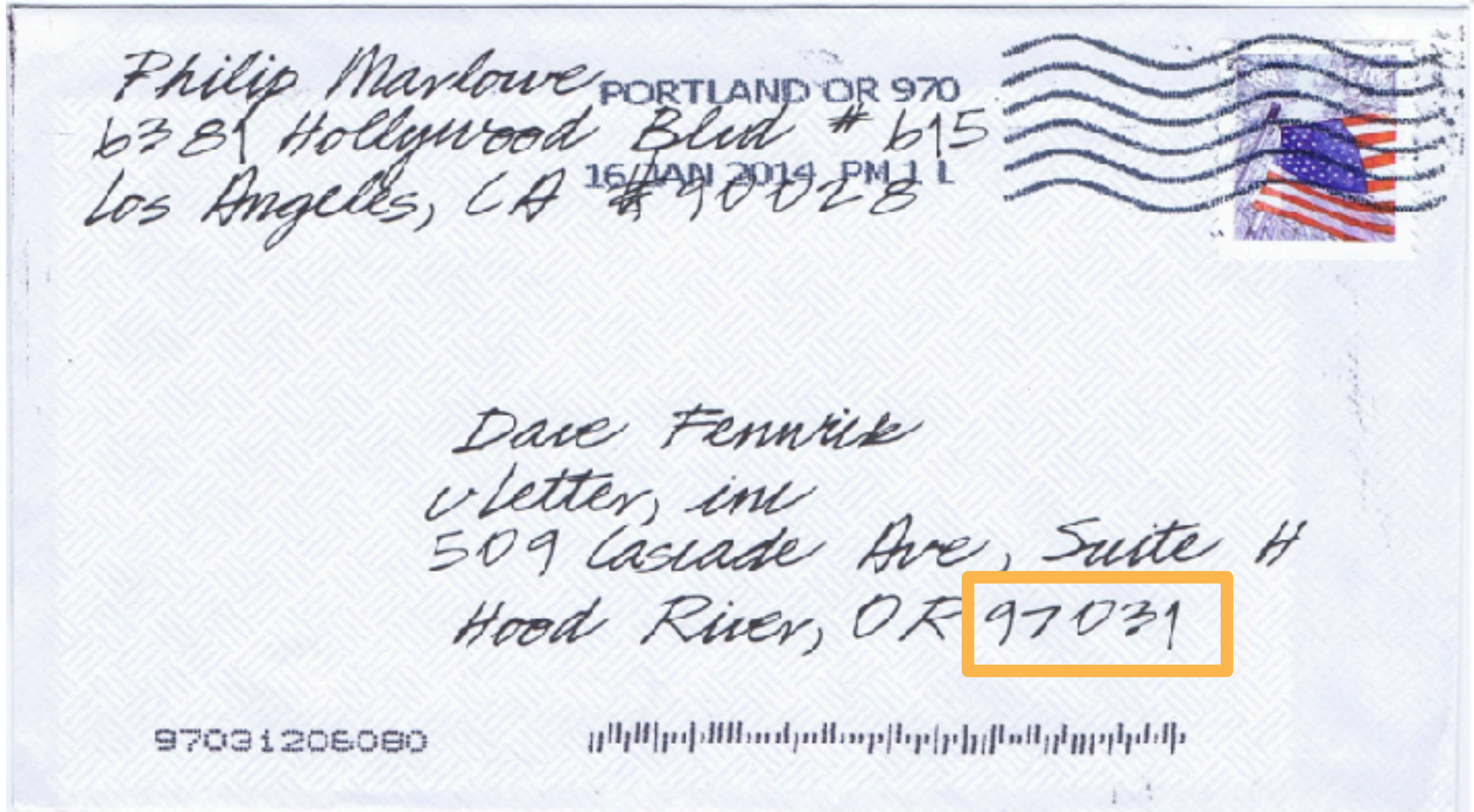


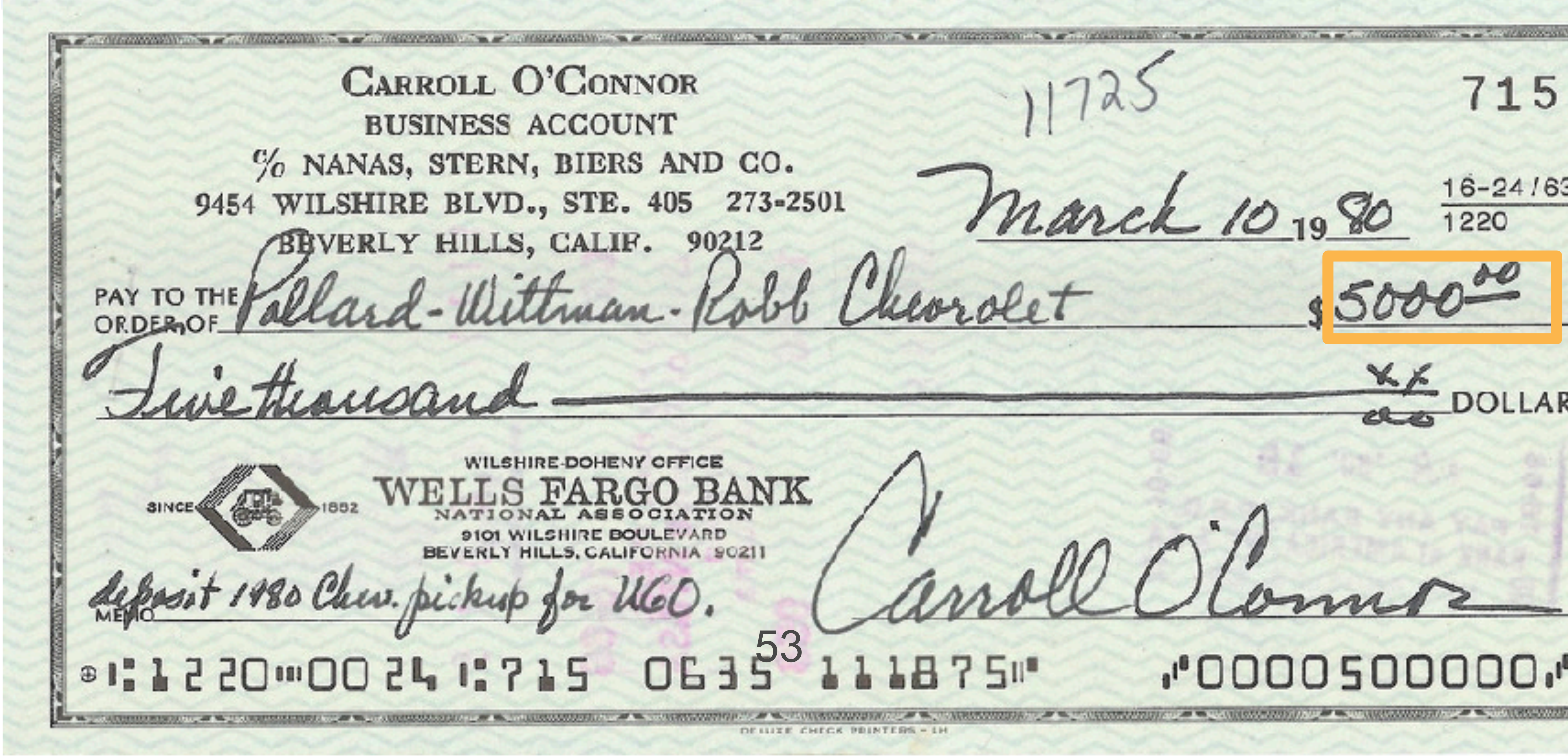


LeNet Architecture

gluon-cv.mxnet.io

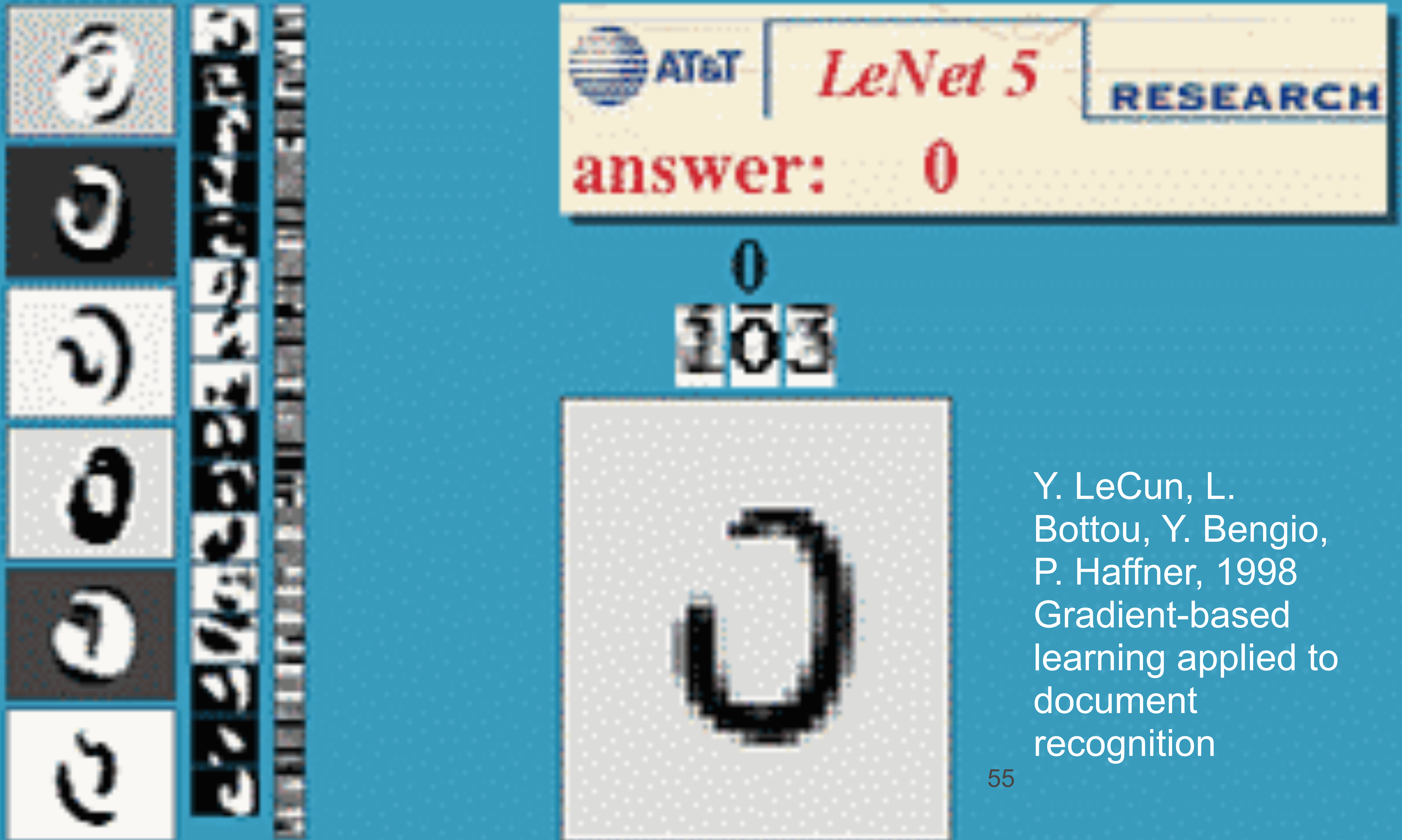
Handwritten Digit Recognition





MNIST

- Centered and scaled
- 50,000 training data
- 10,000 test data
- 28 x 28 images
- 10 classes



Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, 1998 Gradient-based learning applied to document recognition

AlexNet

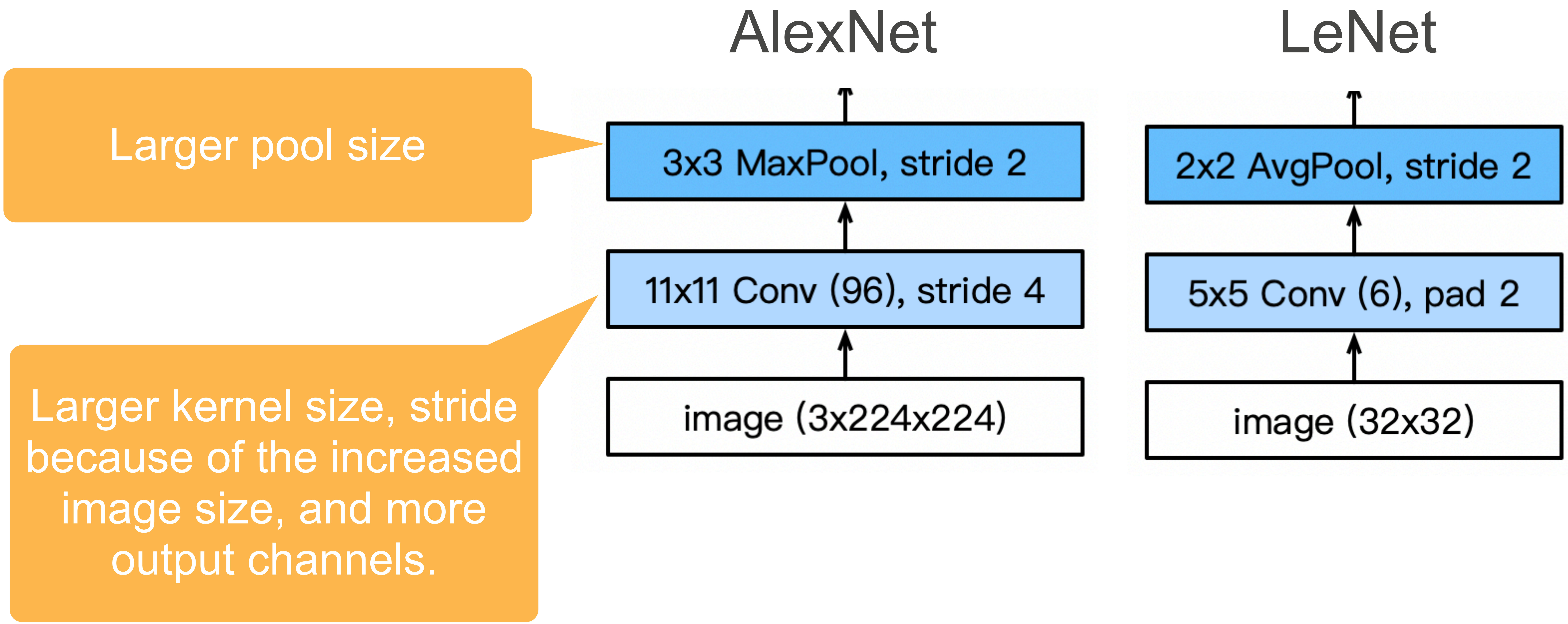
224
224
11
11
11
11
3
Stride of 4
55
55
5
5
5
5
48
48
Max pooling
27
27
3
3
3
3
3
3
3
3
128
128
Max pooling
13
13
13
13
13
13
3
3
3
3
3
3
3
3
192
192
192
192
128
128
Max pooling
dense
dense
dense
2048
2048
2048
2048
1000

Deng et al. 2009

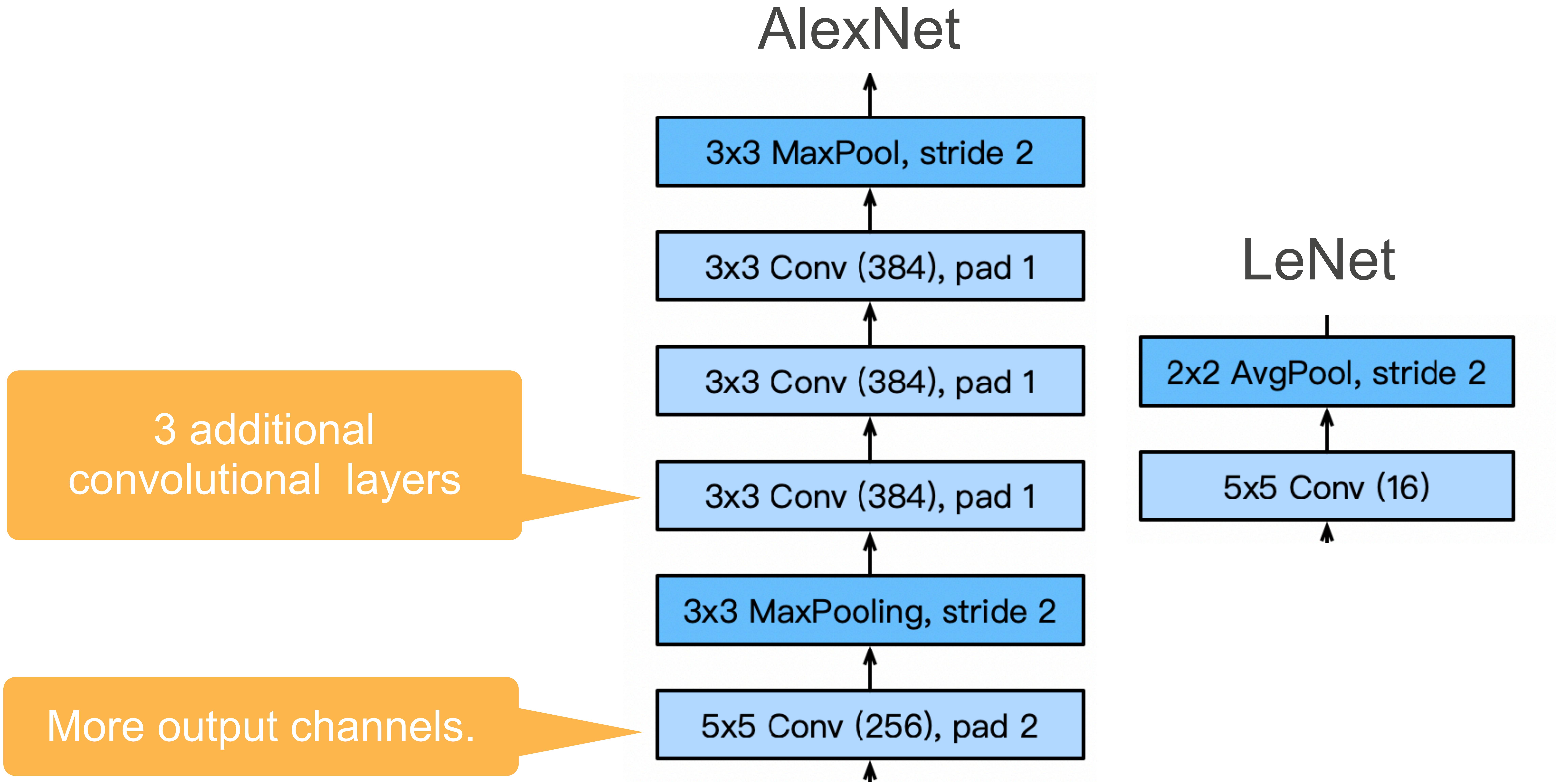
AlexNet

- AlexNet won ImageNet competition in 2012
- Deeper and bigger LeNet
- Paradigm shift for computer vision

AlexNet Architecture



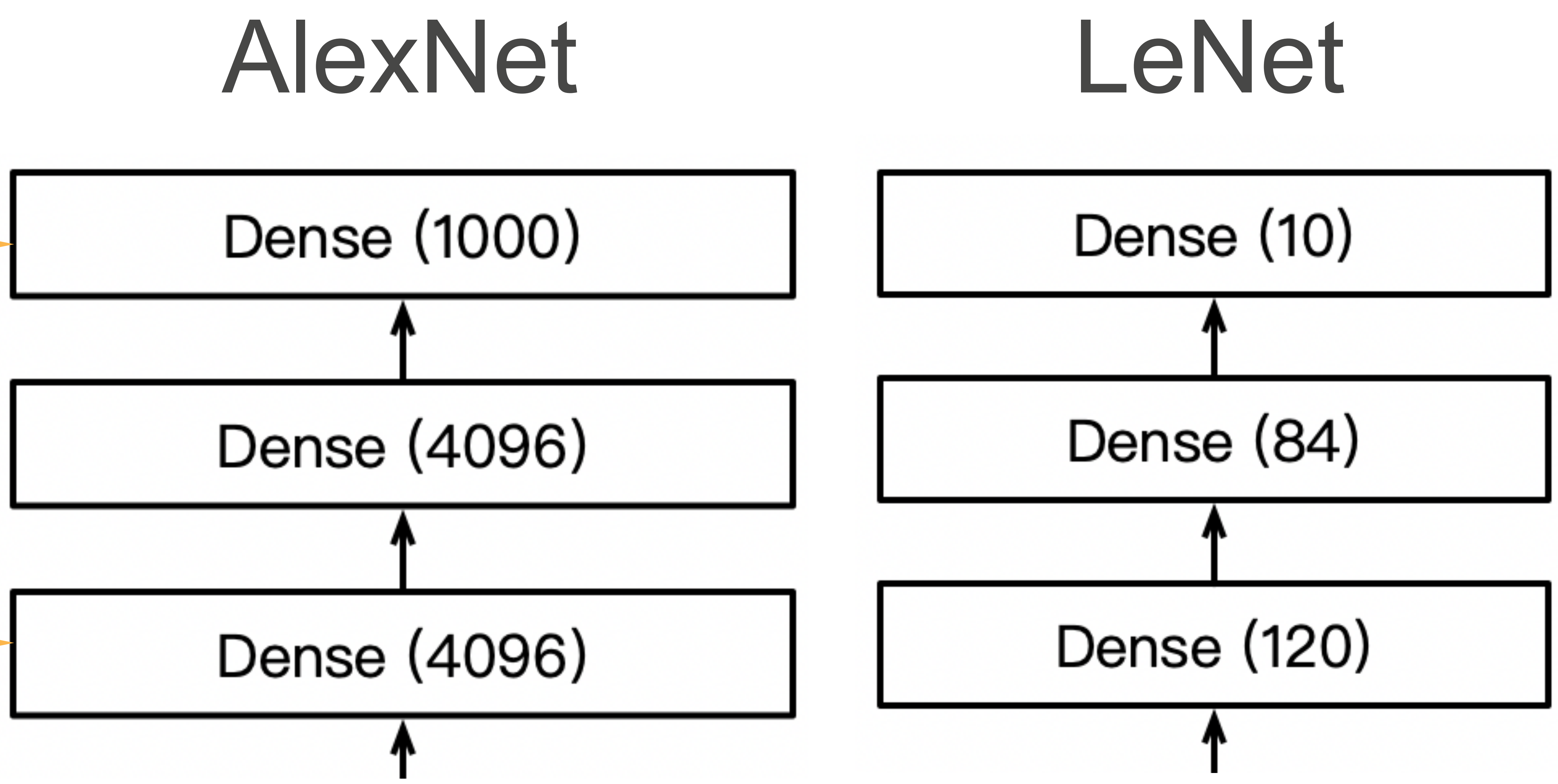
AlexNet Architecture



AlexNet Architecture

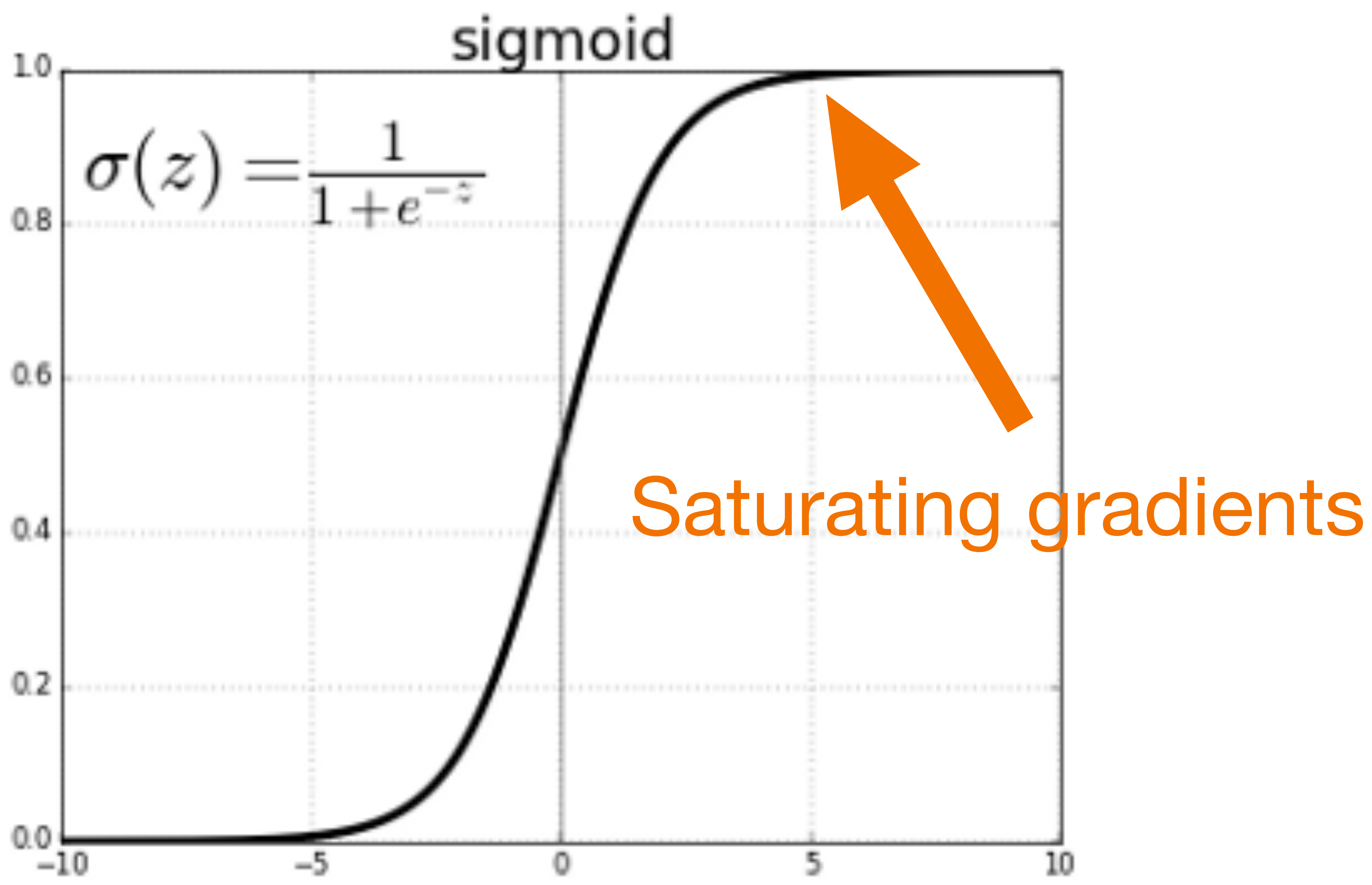
1000 classes output

Increase hidden size from 120 to 4096



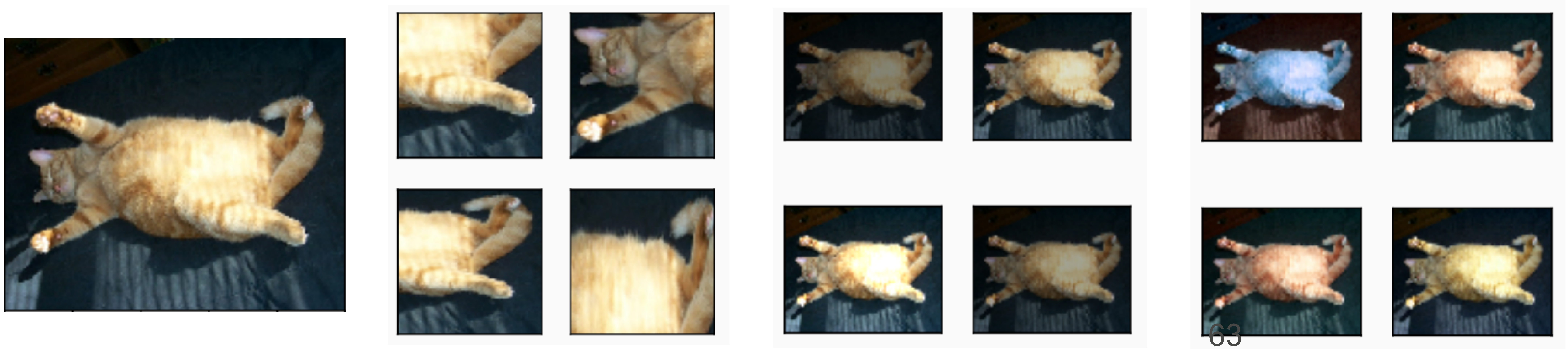
More Differences…

- Change activation function from sigmoid to ReLu (no more vanishing gradient)



More Differences…

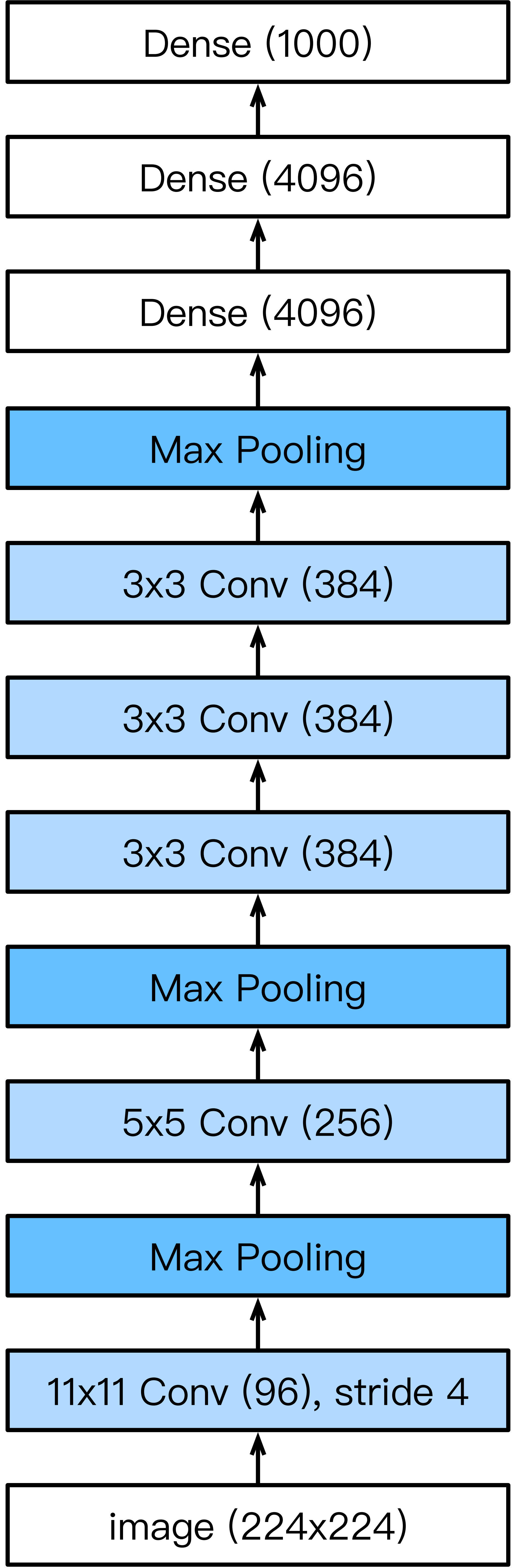
- Change activation function from sigmoid to ReLu (no more vanishing gradient)
- Data augmentation



Complexity

	#parameters	
	AlexNet	**LeNet**
Conv1	35K	150
Conv2	614K	2.4K
Conv3-5	3M	
Dense1	26M	0.048M
Dense2	16M	0.01M
Total	46M	0.06M

X

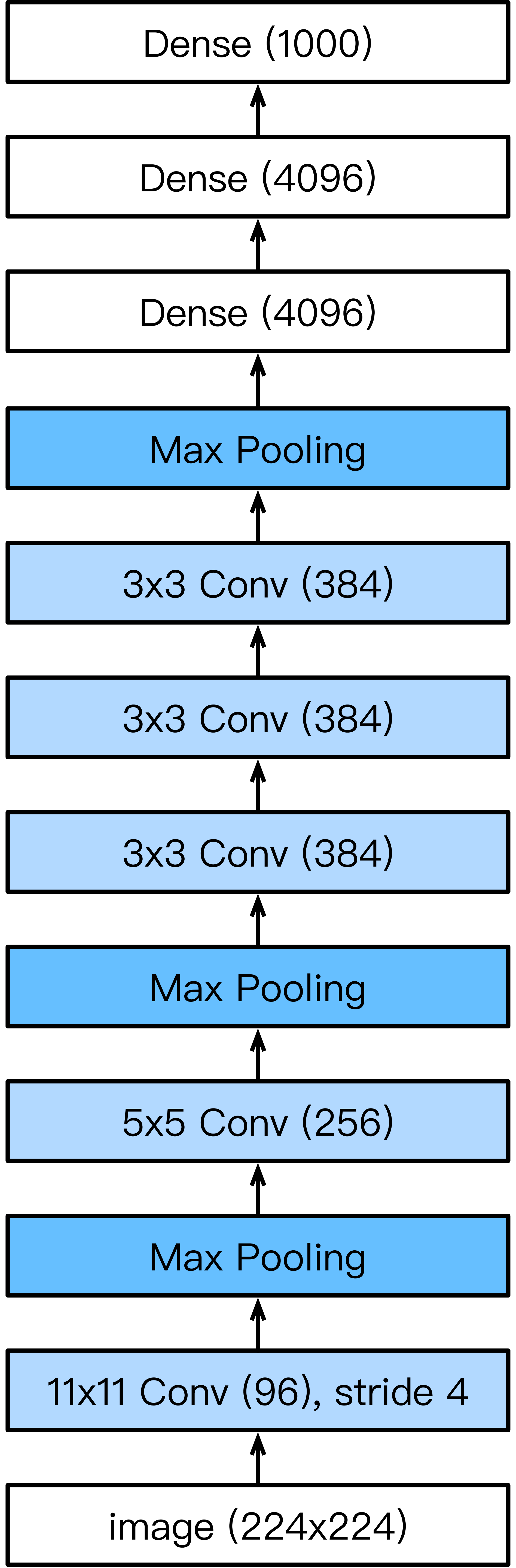


Complexity

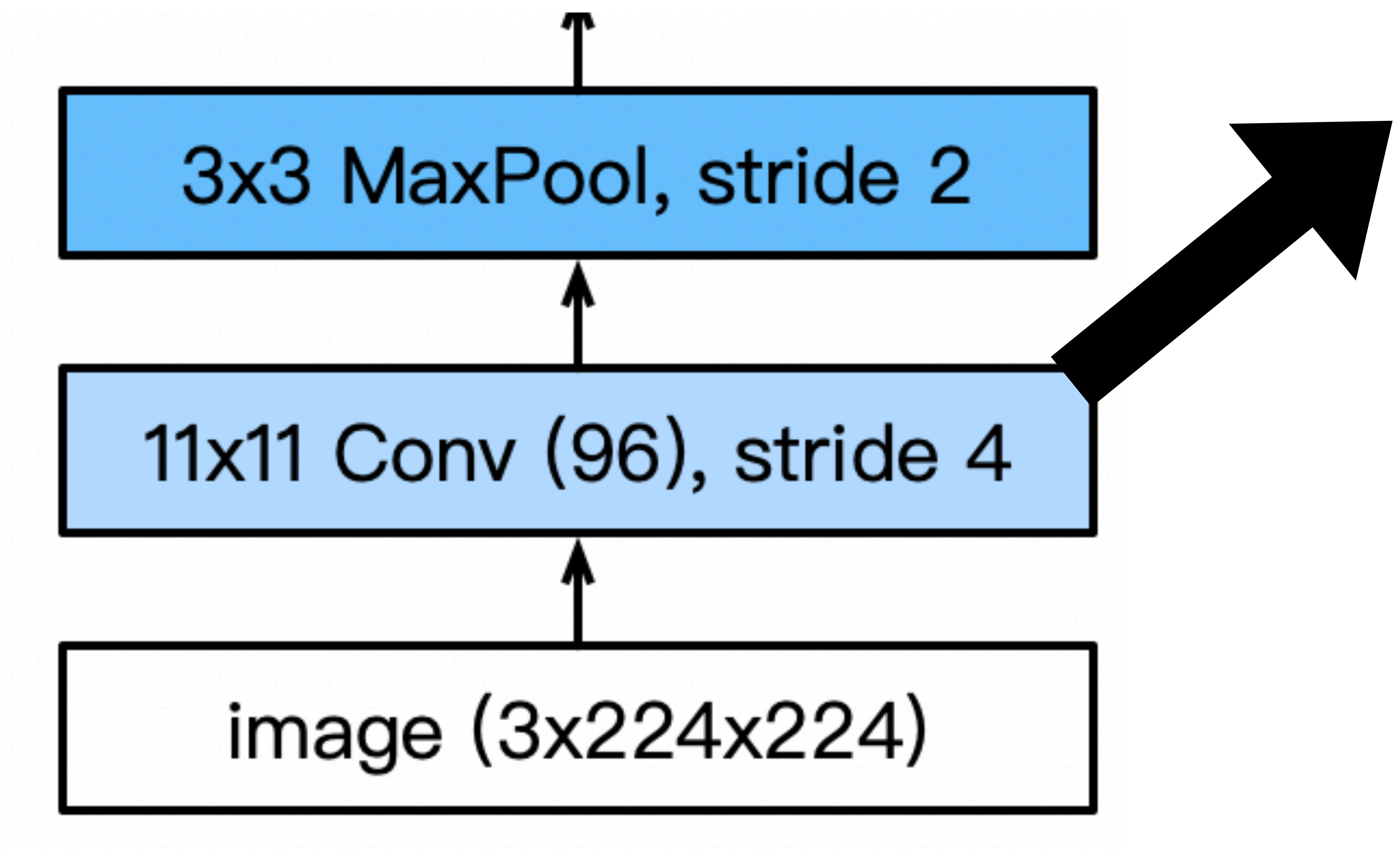
	#parameters	
	AlexNet	LeNet
Conv1	$\boxed{35K}$	150
Conv2	614K	2.4K
Conv3-5	3M	
Dense1	26M	0.048M
Dense2	16M	0.01M
Total	46M	0.06M

11x11x3x96=35k

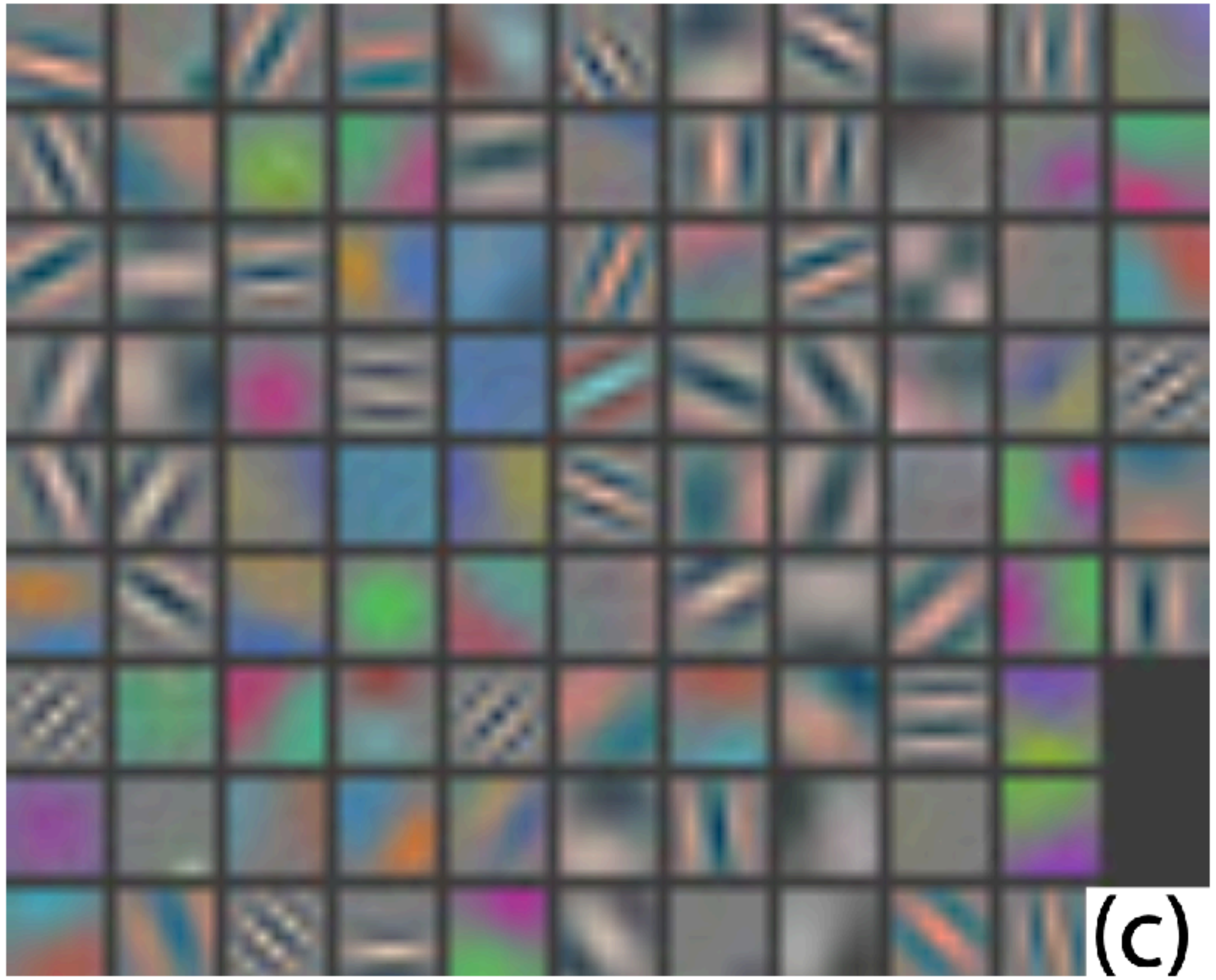
X



AlexNet



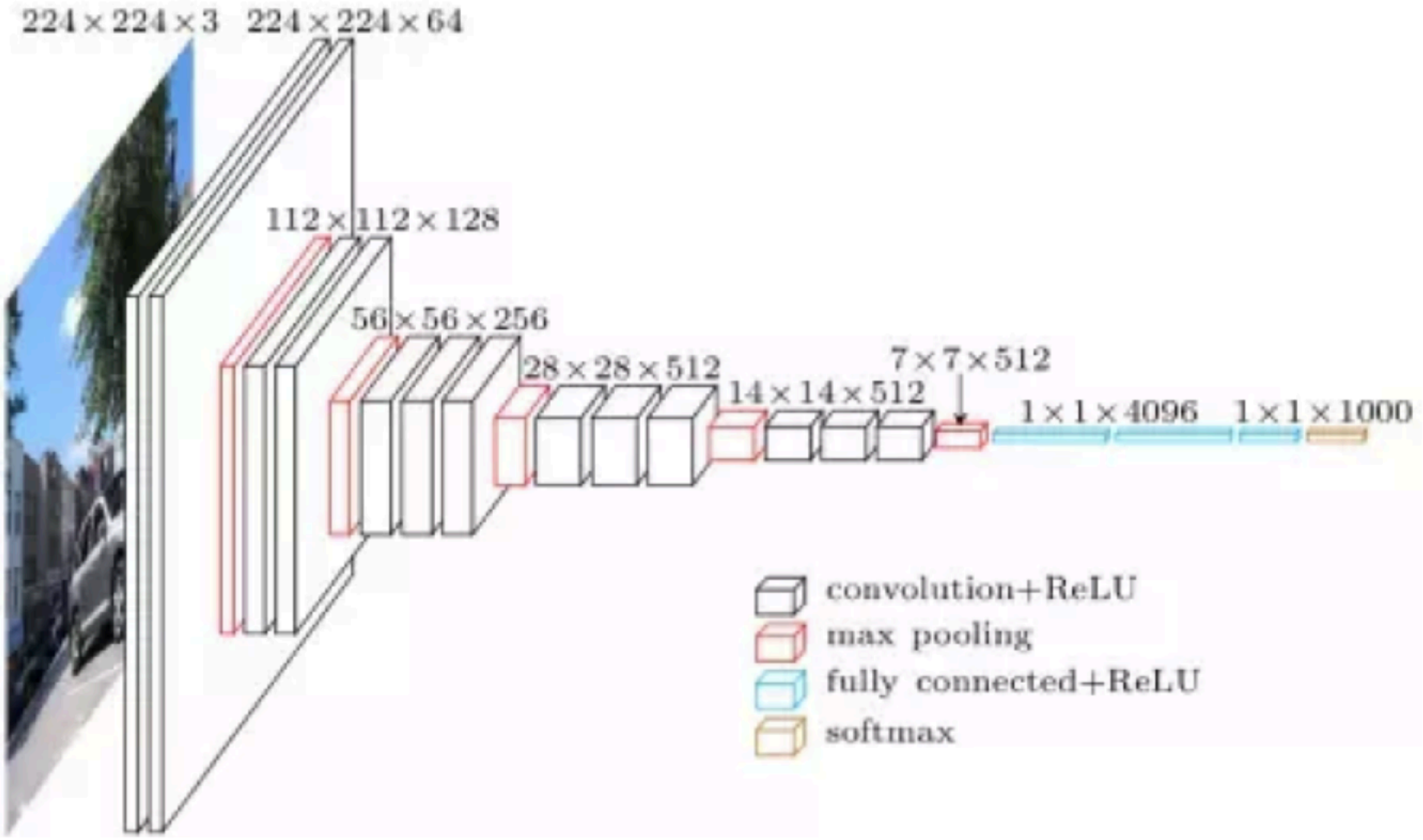
Each Conv1 kernel is 3x11x11, can be visualized as an RGB patch:



[Visualizing and Understanding Convolutional Networks. M Zeiler & R Fergus 2013]

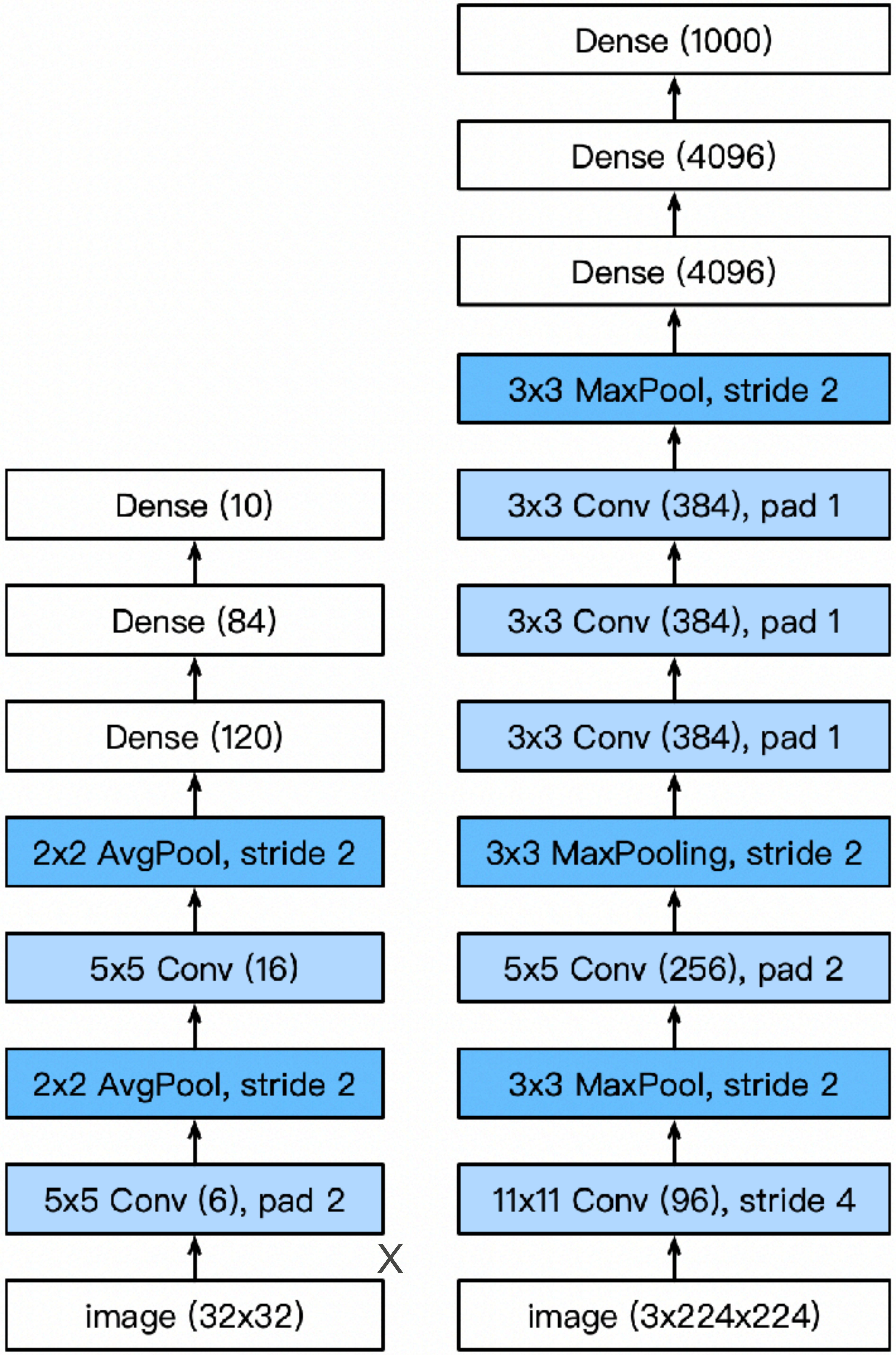
x

VGG


224 × 224 × 3
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU
softmax

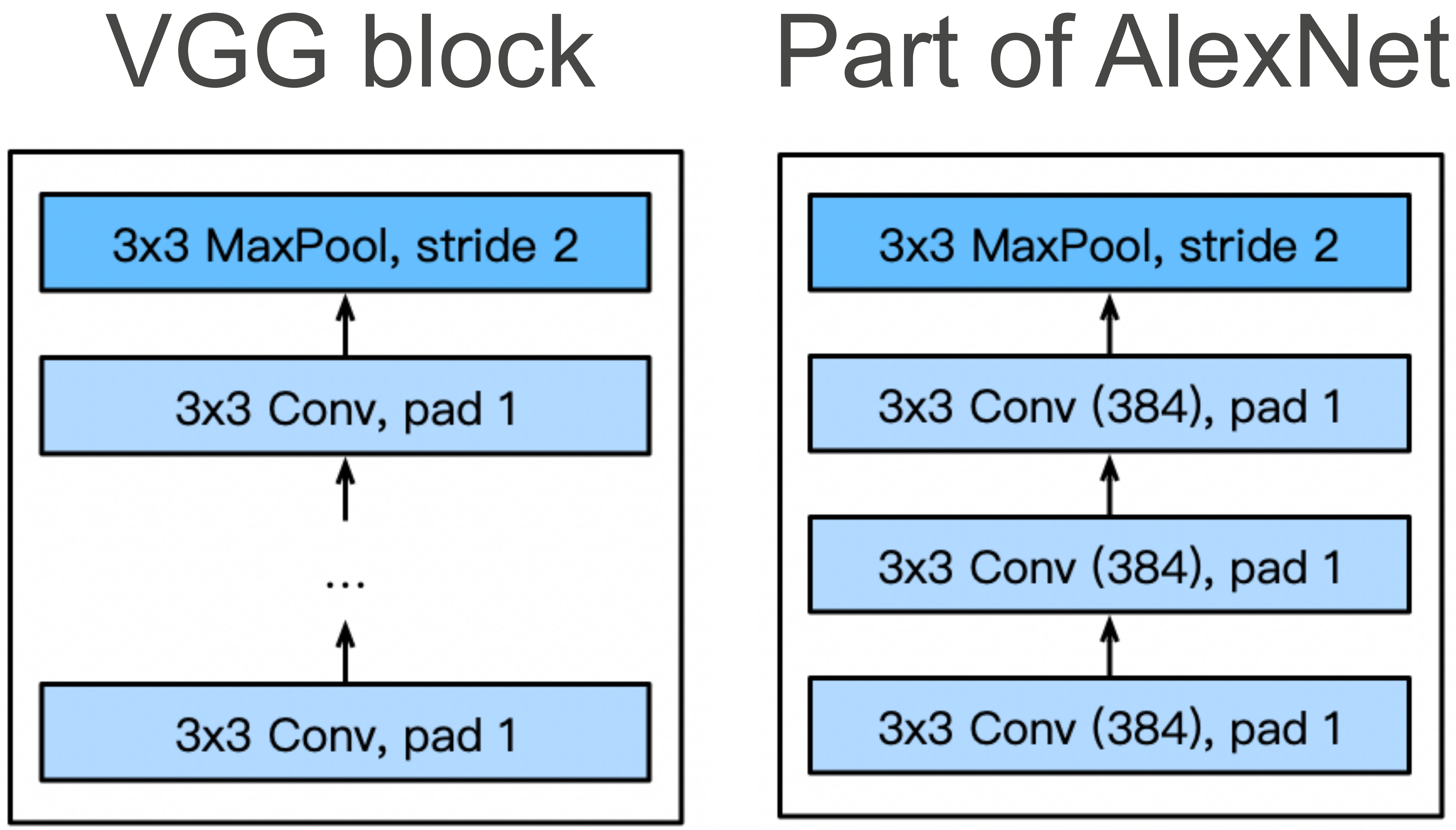
VGG

- AlexNet is deeper and bigger than LeNet to get performance
- Go even bigger & deeper?
- Options
  - More dense layers (too expensive)
  - **More** convolutions
  - Group into **blocks**



VGG Blocks

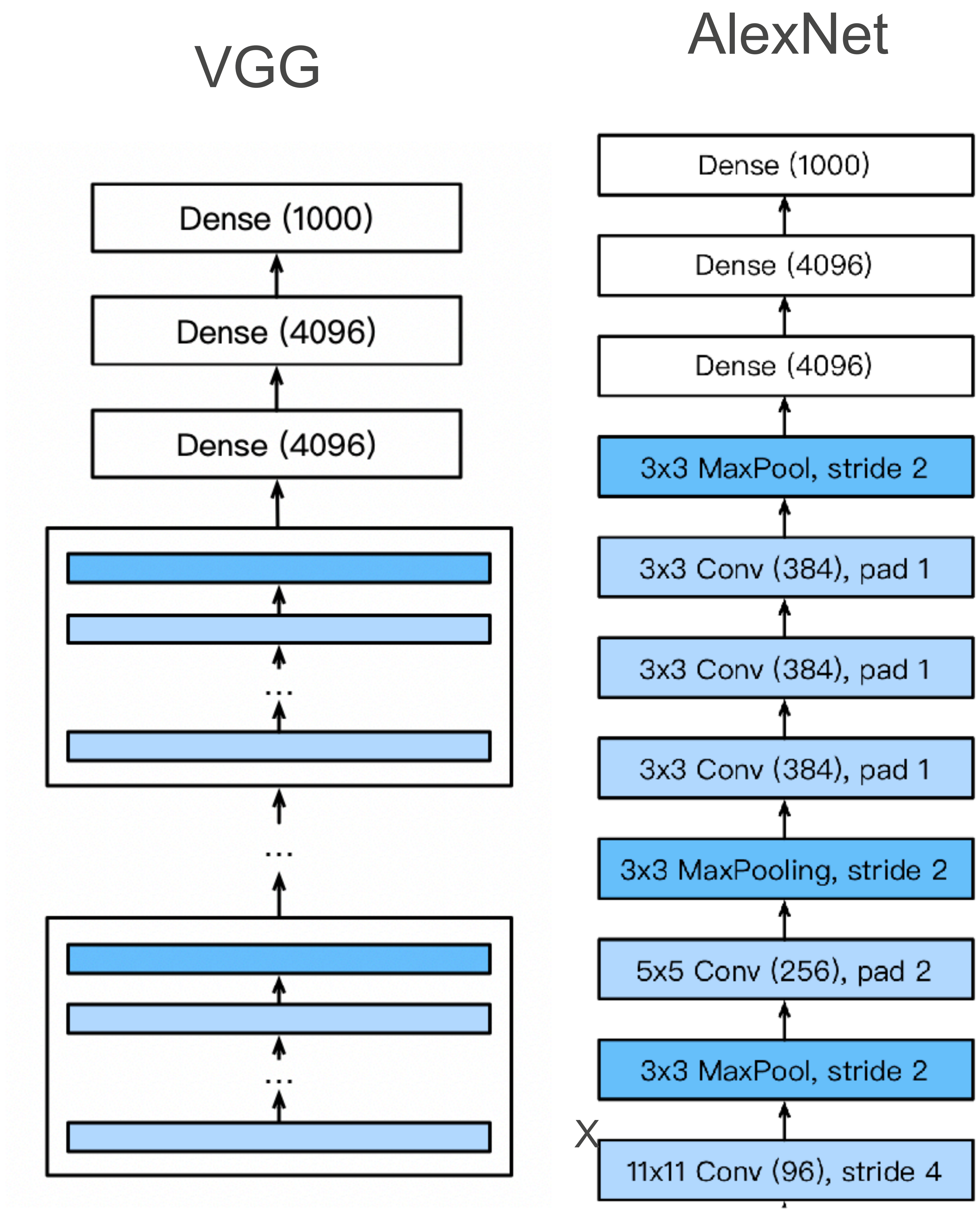
- Deeper vs. wider?
  - 5x5 convolutions
  - 3x3 convolutions (more)
  - **Deep & narrow better**
- VGG block
  - *3x3* convolutions (pad 1) **(n layers, m channels)**
  - 2x2 max-pooling (stride 2)



X

VGG Architecture

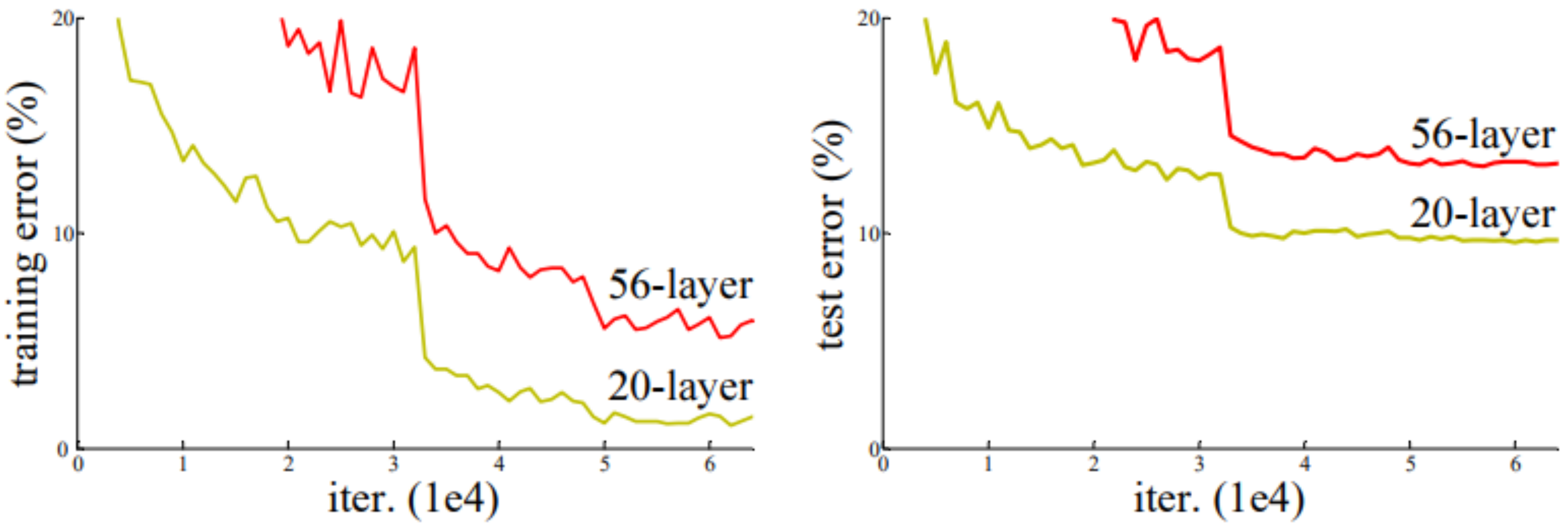
- Multiple VGG blocks followed by dense layers
- Vary the repeating number to get different architectures, such as VGG-16, VGG-19, …



Can we keep adding more layers?

- No! Some problems:
  - Vanishing gradients: more layers ➔ more likely
  - Deeper models are harder to optimize

Reflected in training error:

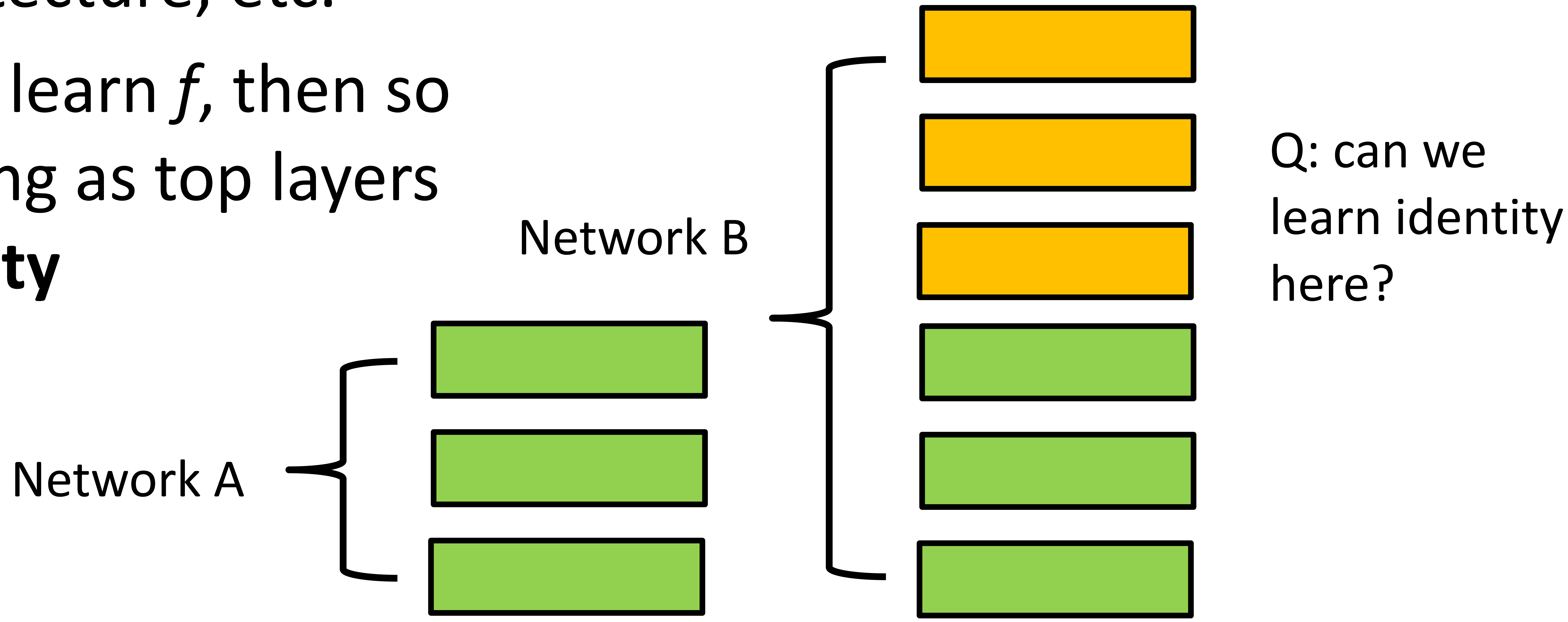


He et al: "Deep Residual Learning for Image Recognition"

Depth Issues & Learning Identity

Why would more layers result in **worse** performance?

- Same architecture, etc.
- If the A can learn *f*, then so can B, as long as top layers learn **identity**

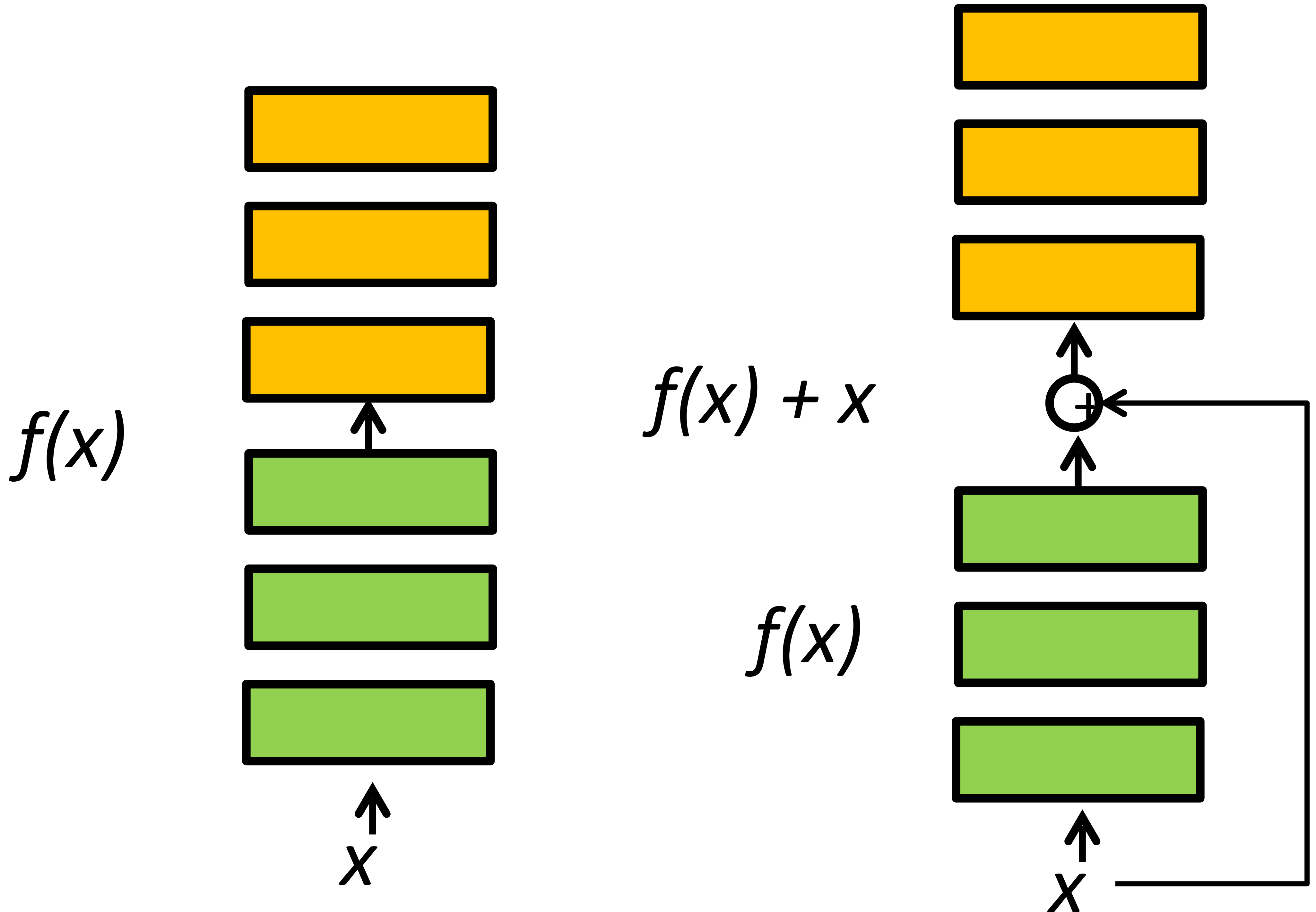


Idea: if layers can learn identity, **can't get** worse

Residual Connections

Idea: Identity might be hard to learn, but zero is easy!

- Make all the weights tiny, produces zero for output
- Can easily transform learning identity to learning zero:



Left: Conventional layers block

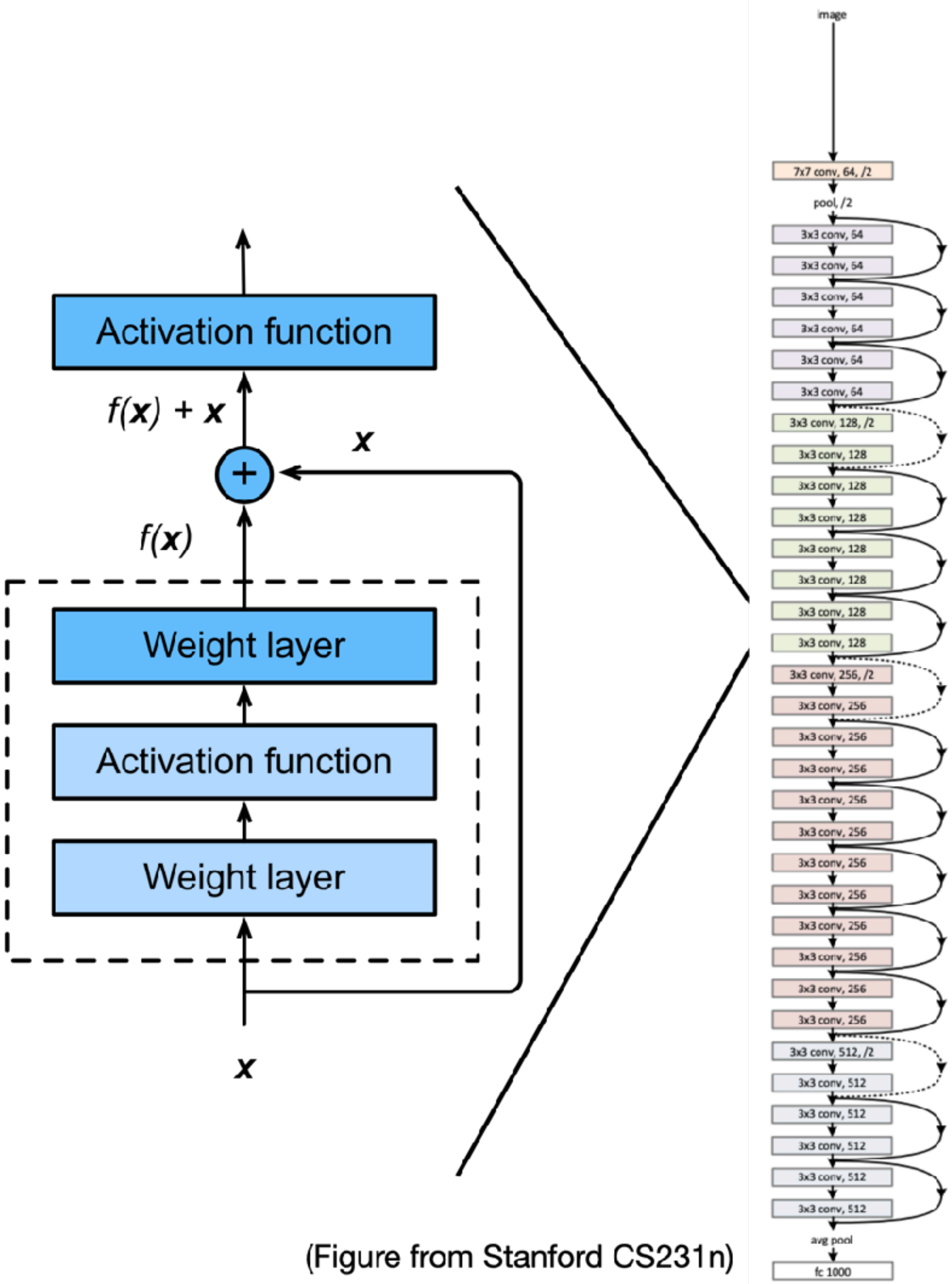
Right: **Residual** layer block

To learn identity *f(x) = x*, layers now need to learn *f(x) = 0* ➔ easier

Full ResNet Architecture

[He et al. 2015]

- Stack residual blocks
- Every residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride of 2 (/2 in each dimension)

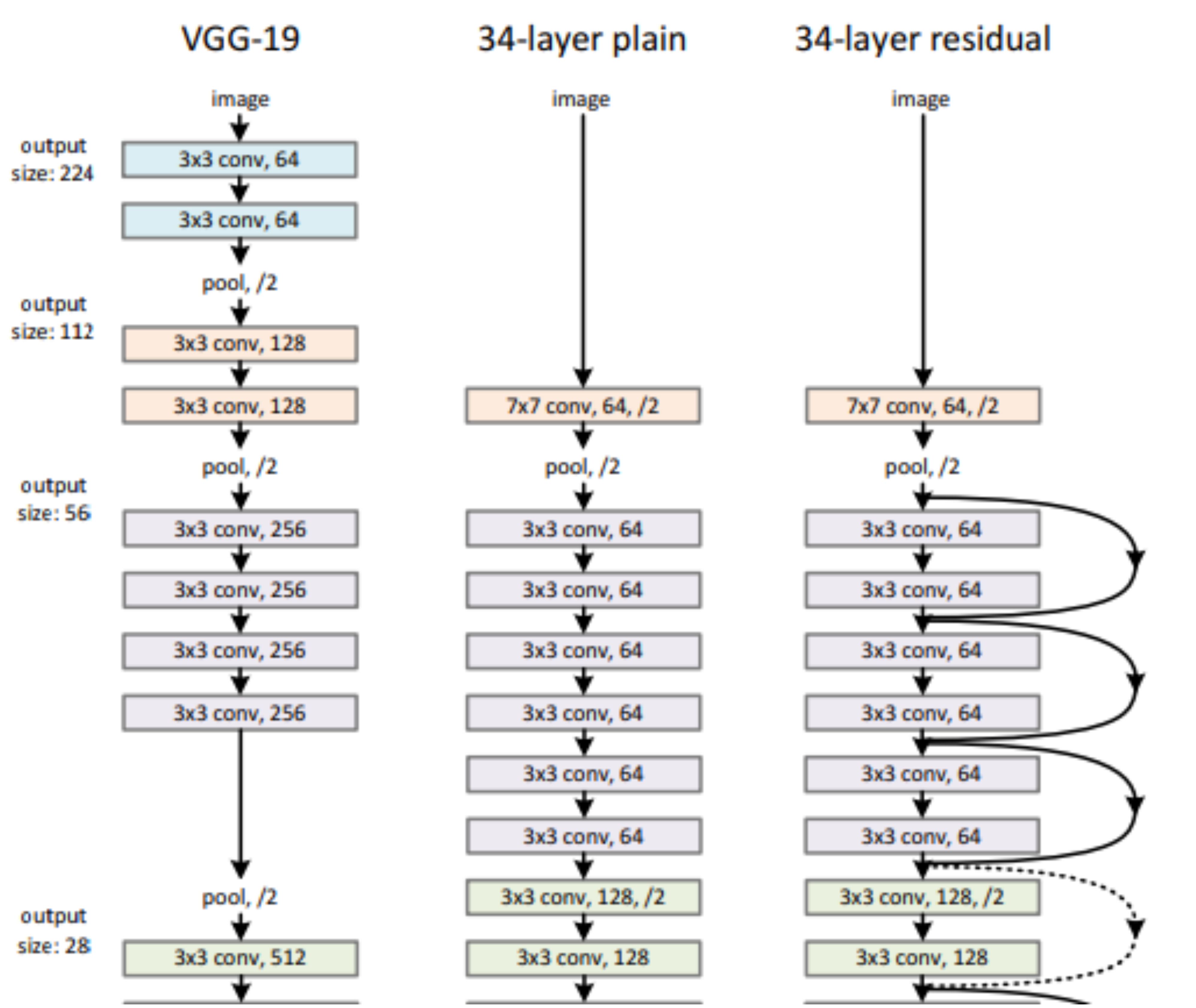


(Figure from Stanford CS231n)

ResNet Architecture

Idea: Residual (skip) connections help make learning easier

- Example architecture:
- Note: residual connections
  – Every two layers for ResNet34
- **Significantly better** performance
  – No additional parameters!
  – Records on many benchmarks



He et al: "Deep Residual Learning for Image Recognition"

ResNet Architecture

Various depth

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Table 1. Architectures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of blocks stacked. Down-sampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

ResNet Architecture

Various depth

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times23$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Table 1. Architectures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of blocks stacked. Down-sampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

ResNet Architecture

Various depth

Repeat x3 times

# of filters

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		[3×3, 64 3×3, 64] ×2	[3×3, 64 3×3, 64] ×3	[1×1, 64 3×3, 64 1×1, 256] ×3	[1×1, 64 3×3, 64 1×1, 256] ×3	[1×1, 64 3×3, 64 1×1, 256] ×3
conv3_x	28×28	[3×3, 128 3×3, 128] ×2	[3×3, 128 3×3, 128] ×4	[1×1, 128 3×3, 128 1×1, 512] ×4	[1×1, 128 3×3, 128 1×1, 512] ×4	[1×1, 128 3×3, 128 1×1, 512] ×8
conv4_x	14×14	[3×3, 256 3×3, 256] ×2	[3×3, 256 3×3, 256] ×6	[1×1, 256 3×3, 256 1×1, 1024] ×6	[1×1, 256 3×3, 256 1×1, 1024] ×23	[1×1, 256 3×3, 256 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512 3×3, 512] ×2	[3×3, 512 3×3, 512] ×3	[1×1, 512 3×3, 512 1×1, 2048] ×3	[1×1, 512 3×3, 512 1×1, 2048] ×3	[1×1, 512 3×3, 512 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Table 1. Architectures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of blocks stacked. Down-sampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

X

ResNet Architecture

Various depth

1+ 2x3 + 2x4 + 2x6 + 2x3 +1 = 34

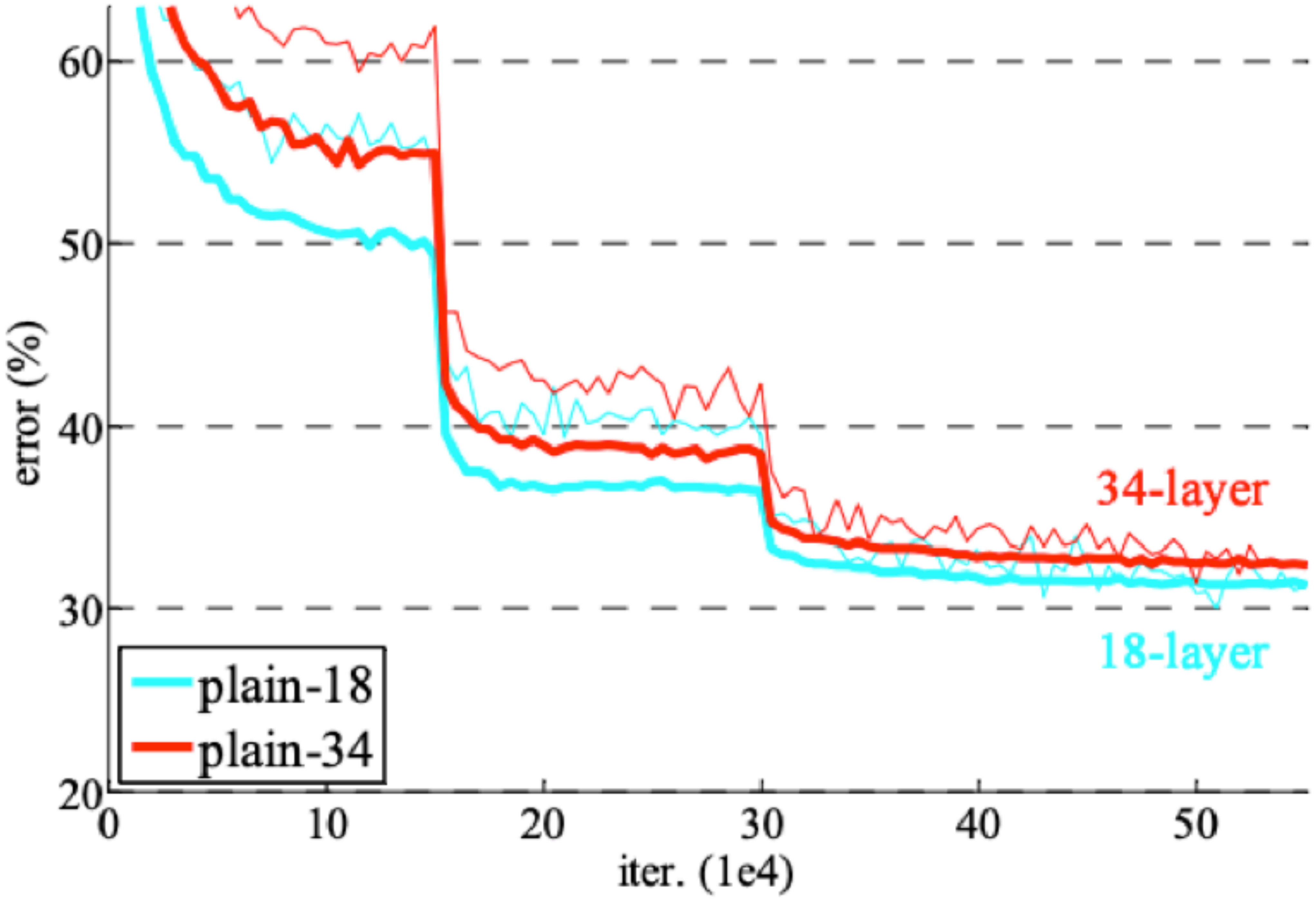
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times23$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

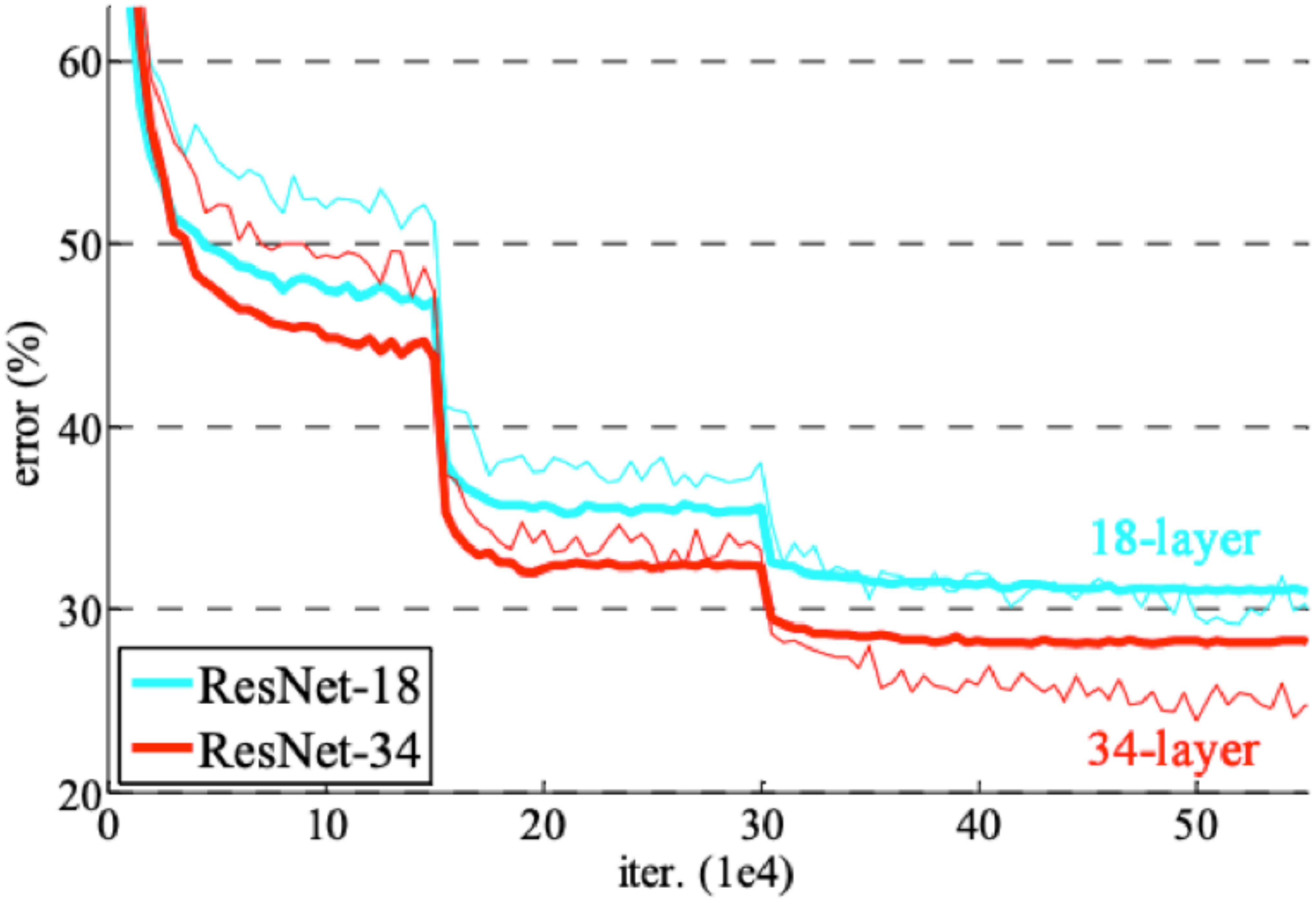
Table 1. Architectures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of blocks stacked. Down-sampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

X

ResNet Training Curves on ImageNet

[He et al., 2015]





A Bit More on ResNets

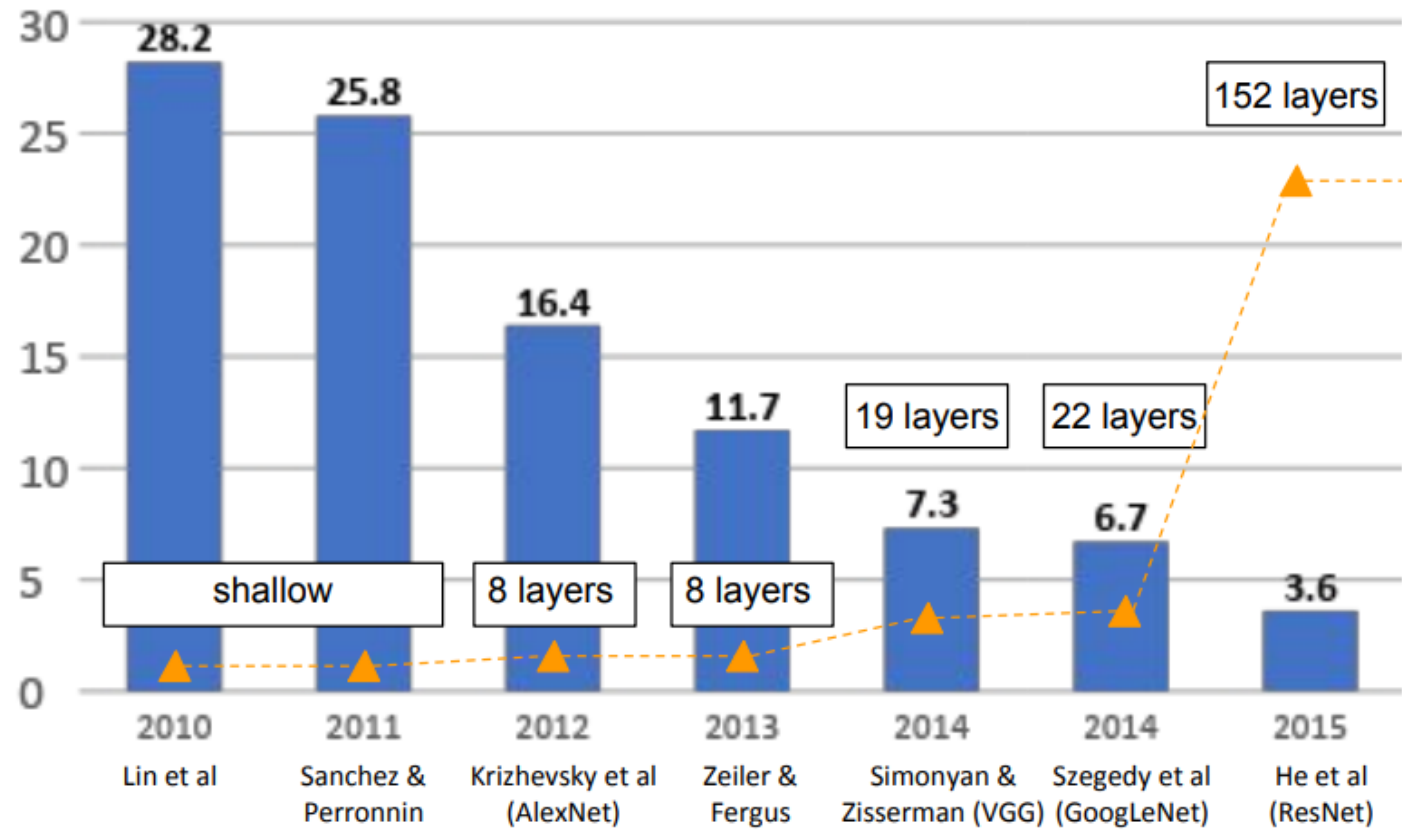
Idea: Residual (skip) connections help make learning easier

- Note: Can also analyze from **backpropagation** p.o.v
  - Residual connections add paths to computation graph
- Also uses **batch normalization**
  - Normalize the features at each layer to have same mean/variance
  - Common deep learning trick
- Highway networks: learn weights for residual connections

Ioffe and Szegedy: “Batch Normalization: Accelerating Deep Network Training b y Reducing Internal Covariate Shift”

Evolution of CNNs

ImageNet competition (error rate)



Credit: Stanford CS 231n

Acknowledgement:
Some of the slides in these lectures have been adapted from materials developed by Alex Smola and Mu Li:
https://courses.d2l.ai/berkeley-stat-157/index.html